Oliver Kreipl, Christian Eichhorn, Dorian Karnbaum, Heiko Marr †

Linux-Systemadministration

Ein Webmasters Press Lernbuch

Version 1.3.0 vom 03.06.2019

Autorisiertes Curriculum für das Webmasters Europe Ausbildungs- und Zertifizierungsprogramm

Inhaltsverzeichnis

Berechtigungssystem

8

In dieser Lektion lernen Sie

- wie das Rechtesystem unter Linux funktioniert.
- wie Sie den Besitzer und die Gruppen für Dateien und Verzsichnisse setzen können.
- welche speziellen Rechte es gibt und wie sie eingesetzt werden.

8.1 Einführung

Dieser Kurs setzt voraus, dass Sie wissen, wie Sie als unprivilegierter Benutzer mit dem Befehl `chmod` Schreib-, Lese- und Ausführen-Rechte auf Ihre eigenen Dateien und Verzeichnisse vergeben. Wir wollen uns an dieser Stelle ansehen, wie Sie als Benutzer *root* die Gruppe und den Besitzer von Dateien und Verzeichnissen ändern können. Das gibt Ihnen viele Freiheiten, bringt aber auch eine große Verantwortung mit sich. Seien Sie sich dessen immer bewußt und ändern Sie Rechte auf Dateien nur dann, wenn Sie genau wissen, was Sie tun.

Für eine Datei oder ein Verzeichnis kann jeweils nur **ein** Besitzer und **eine** Gruppe angegeben werden. Wenn Sie also möchten, dass bestimmte Benutzer spezielle Rechte auf eine Datei oder ein Verzeichnis haben, müssen diese der gleichen Gruppe angehören.

Gruppen können nicht anderen Gruppen zugeordnet werden.

8.2 Besitzer und Gruppe mit chown festlegen

Bevor Sie beginnen, Rechte zu verteilen, sollten Sie festlegen, für wen diese Rechte gültig sind. Sobald eine Datei erstellt wird, bekommt sie als Besitzer den Benutzernamen und als Gruppe die Hauptgruppe des aktuellen Benutzers zugewiesen.

Beispiel

```
root@frodo:~# touch /var/tmp/permissions.txt

root@frodo:~# ls -l /var/tmp/permissions.txt
-rw-r--r-- 1 root root 0 Jul 11 10:50 /var/tmp/permissions.txt
```

Sie können als Besitzer der Datei oder als *root* einen anderen Besitzer bzw. eine andere Gruppe zuweisen. Dazu verwenden Sie den Befehl `chown`. Zunächst wieder die Syntax:

```
chown [OPTION]... [BESITZER][:GRUPPE] DATEI...
```

Nehmen wir zuerst den einfachsten Fall: Sie möchten den Besitzer der Datei *permissions.txt* ändern. Zu diesem Zweck geben Sie nur den Benutzernamen an, der als Besitzer der Datei gültig sein soll:

Beispiel

```
root@frodo:~# chown customer1 /var/tmp/permissions.txt
```

```
root@frodo:~# ls -l /var/tmp/permissions.txt
-rw-r--r-- 1 customer1 root 0 Jul 11 10:50 /var/tmp/permissions.txt
```

Somit treffen die gesetzten Berechtigungen des Besitzers jetzt auf *customer1* zu. Dementsprechend darf *customer1* die Datei lesen und schreiben.

Der Befehl `chown` bietet aber noch mehr, als nur den Besitzer zu bestimmen. Sie können gleichzeitig eine Gruppe angeben, die ebenfalls gesetzt wird. Folgendes Beispiel macht es deutlich:

Beispiel

```
root@frodo:~# chown customer2:customers /var/tmp/permissions.txt

root@frodo:~# ls -l /var/tmp/permissions.txt
-rw-r--r-- 1 customer2 customers 0 Jul 11 10:50 /var/tmp/permissions.txt
```

Wenn Sie Dateien einem Benutzer und dessen Hauptgruppe zuordnen wollen, dann müssen Sie die Gruppe nicht angeben, sondern es genügt, den Doppelpunkt (`:`) zu setzen.

Beispiel

```
root@frodo:~# chown customer1: /var/tmp/permissions.txt

root@frodo:~# ls -l /var/tmp/permissions.txt
-rw-r--r-- 1 customer1 customer1 0 Jul 11 10:50 /var/tmp/permissions.txt
```

Möchten Sie nur die Gruppe ändern, können Sie dies auf zwei verschiedene Arten tun. Der Standardweg geht über den Befehl `chgrp`.

Beispiel

```
root@frodo:~# chgrp customers /var/tmp/permissions.txt

root@frodo:~# ls -l /var/tmp/permissions.txt
-rw-r--r-- 1 customer1 customers 0 Jul 11 10:50 /var/tmp/permissions.txt
```

Die zweite Art besteht darin, den Befehl `chown` zu verwenden. Auch damit können Sie die Gruppe allein ändern. Dazu lassen Sie einfach den Besitzer weg. Wichtig dabei ist, dass der Doppelpunkt der Gruppe vorangestellt wird:

Beispiel

```
root@frodo:~# chown :employees /var/tmp/permissions.txt

root@frodo:~# ls -l /var/tmp/permissions.txt
-rw-r--r-- 1 customer1 employees 0 Jul 11 10:50 /var/tmp/permissions.txt
```

Da unter Linux alles als Datei behandelt wird, können Sie auch die Besitzer und Gruppen von Verzeichnissen auf die gleiche Art und Weise verändern.

Beachten Sie, dass beim Ändern der Besitzrechte eines Verzeichnisses nur die des Verzeichnisses selbst geändert werden.

Die Dateien, die innerhalb des Verzeichnisses liegen, haben nach wie vor ihren bisherigen Besitzer und ihre Gruppe.

Um die Rechte auf ein Verzeichnis samt Inhalt und Unterverzeichnissen zu ändern, müssen Sie den jeweiligen Befehl (`chown` oder `chgrp`) rekursiv anwenden. Dies geschieht mit der Option `-R`.

Im Beispiel weisen wir unser komplettes Homeverzeichnis (*/root*) der Gruppe *employees* zu.

Beispiel

```
root@frodo:~# chown -R :employees ~

root@frodo:~# ls -la ~
total 28
drwx------  2 root employees 4096 Jul 22 15:59 .
drwxr-xr-x 22 root root      4096 Jul  4 14:07 ..
-rw-------  1 root employees 2804 Sep  2 16:26 .bash_history
-rw-r--r--  1 root employees  570 Jan 31  2010 .bashrc
-rw-r--r--  1 root employees  140 Nov 19  2007 .profile
-rw-------  1 root employees 1287 Jul 22 15:59 .viminfo
-rw-r--r--  1 root employees   65 Jul  5 12:36 .vimrc
```

Sie sehen, dass das Verzeichnis selbst sowie alle darin enthaltenen Dateien und Verzeichnisse jetzt der Gruppe *employees* gehören. Da ich nicht möchte, dass alle Mitglieder der Gruppe *employees* auf die Dateien im Homeverzeichnis von *root* zugreifen können, mache ich die Änderung natürlich wieder rückgängig, diesmal mit dem Befehl `chgrp -R`:

Beispiel

```
root@frodo:~# chgrp -R root ~

root@frodo:~# ls -la ~
total 28
drwx------  2 root root 4096 Jul 22 15:59 .
drwxr-xr-x 22 root root 4096 Jul  4 14:07 ..
-rw-------  1 root root 2804 Sep  2 16:26 .bash_history
-rw-r--r--  1 root root  570 Jan 31  2010 .bashrc
-rw-r--r--  1 root root  140 Nov 19  2007 .profile
-rw-------  1 root root 1287 Jul 22 15:59 .viminfo
-rw-r--r--  1 root root   65 Jul  5 12:36 .vimrc
```

Übung 16:

1. Erstellen Sie die Datei */var/tmp/exercise_permissions.txt*.
2. Ändern Sie den Besitzer der Datei auf *marco*. Falls der Benutzer auf Ihrem System noch nicht existiert: legen Sie ihn bitte an und geben Sie ihm ein Homeverzeichnis.
3. Ändern Sie die Gruppe des Homeverzeichnisses von *marco* inkl. aller Dateien und Unterverzeichnissen auf *customers*. Verwenden Sie dazu den Befehl `chgrp`.

4 Ändern Sie den Besitzer des kompletten Homeverzeichnisses von *marco* auf *marco* und dessen Hauptgruppe.

8.3 Spezielle Rechte

Unter Linux gibt es nicht nur die Standardrechte, die Sie bisher in dieser Lektion gelernt haben, sondern auch noch spezielle Rechte, die für Dateien und Verzeichnisse gelten können. Nehmen wir z. B. das Verzeichnis */tmp*.

Der Unterschied zwischen */tmp* und */var/tmp* besteht übrigens darin, das */tmp* bei jedem Neustart gelöscht wird. Die Dateien in */var/tmp* sind auch nach einem Neustart noch vorhanden.

Dieses Verzeichnis steht sämtlichen Benutzern und Programmen zur Zwischenspeicherung zur Verfügung. Aus diesem Grund müssen für alle Benutzer Lese- und Schreibrechte sowie das Recht, das Verzeichnis auszuführen (in das Verzeichnis zu wechseln), gegeben sein. Würden wir unsere Standardrechte anwenden, die wir kennen, würde ein herkömmlicher Benutzer die temporären Dateien von *root* löschen können. Linux hat genau für dieses Problem eine Lösung parat, das sog. »Sticky Bit«.

8.3.1 Sticky Bit

Ein Bit ist die kleinste Einheit in einem Rechner, die einen Zustand annehmen kann. Also entweder 1 (An) oder 0 (Aus). Das binäre System lässt grüßen.

Bei unseren Rechten arbeiten wir ebenfalls mit Bits. Jedes Recht hat einen Zustand — es ist also ein Bit gesetzt oder nicht.

Das Sticky Bit wenden Sie auf Verzeichnisse an. Sie teilen dem System damit mit, dass eine Datei innerhalb dieses Verzeichnisses nur von seinem Besitzer geändert werden kann.

Dies löst also unser Problem mit dem Verzeichnis */tmp*. Wenn Sie sich die Rechte des Verzeichnisses anzeigen lassen, werden Sie das Sticky Bit erkennen.

Beispiel

```
root@frodo:~# ls -ld /tmp
drwxrwxrwt 2 root root 4096 Jul 14 11:50 /tmp
```

Das Sticky Bit wird durch das `t` repräsentiert und ersetzt im dritten Block das `x`.

Setzen können Sie das Sticky Bit auf zwei verschiedene Arten, je nachdem, welche Schreibweise Sie bevorzugen:

Beispiel

Attributmodus

```
root@frodo:~# chmod o=rwxt /var/tmp/permissions_directory
```

Beispiel

Oktalschreibweise

```
root@frodo:~# chmod 1777 /var/tmp/permissions_directory
```

Beachten Sie, dass im Attributmodus das Sticky Bit auf »`o`thers« (alle anderen Benutzer) gesetzt werden muss.

Entfernen können Sie das Sticky Bit ebenfalls auf zwei Arten:

Beispiel

Attributmodus

```
root@frodo:~# chmod o-t /var/tmp/permissions_directory
```

Beispiel

Oktalschreibweise

```
root@frodo:~# chmod 0777 /var/tmp/permissions_directory
```

Übung 17:

1. Erstellen Sie das Verzeichnis */var/tmp/permissions_directory*, falls noch nicht geschehen.
2. Vergeben Sie für *others* die Berechtigungen `lesen`, `schreiben` und `ausführen` auf das Verzeichnis. Setzen Sie außerdem das Sticky Bit. Nutzen Sie dazu die Attributschreibweise.
3. Wechseln Sie Ihre Identität zu *customer2* und erstellen Sie eine Datei in diesem Verzeichnis.
4. Wechseln Sie Ihre Identität zu *customer1* und versuchen Sie, die Datei zu löschen.

8.3.2 SUID-Bit

Wenn Sie unter Linux ein Programm ausführen, weiß Linux, dass Sie dieses Programm gestartet haben. Es merkt sich dabei den Benutzer, als der Sie im System unterwegs sind und lässt das Programm mit den Rechten dieses Benutzers laufen. Das Programm läuft also unter Ihrem Namen und mit Ihren Rechten. Sind Sie *root*, ist alles in Ordnung, denn als *root* dürfen Sie sowieso alles. Für einen normalen Benutzer sieht die Sache anders aus.

Nehmen wir die Datei */etc/shadow*. Wie Sie bereits wissen, werden dort die Passwörter der Benutzer gespeichert. Sehen wir uns die Rechte dieser Datei an:

Beispiel

```
root@frodo:~# ls -l /etc/shadow
-rw-r----- 1 root shadow 1563 Jul 10 15:51 /etc/shadow
```

Sie sehen: schreiben darf in diese Datei nur *root*. Sie erinnern sich noch an das Programm, mit dem Benutzer ihr Passwort ändern können? Richtig: das Programm `passwd`. Jeder Benutzer kann es ausführen, um sein Passwort zu ändern. Folglich müsste das Programm auch mit den Rechten des jeweiligen Benutzers laufen, aber dann könnte das Programm nicht in die Datei */etc/shadow* schreiben, denn dies darf nur *root*.

Das ***SUID-Bit*** (SetUserID-Bit) löst das Problem. Es ist bei der Datei */usr/bin/passwd* bereits gesetzt und wir können es uns ansehen:

Beispiel

```
root@frodo:~# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 34888 Feb 27 2007 /usr/bin/passwd
```

Erkennen können Sie das SUID-Bit an dem `s`, das anstelle des `x` im Rechteblock des Besitzers sitzt.

Sobald das SUID-Bit gesetzt ist, wird das jeweilige Programm oder Skript mit den Rechten des Besitzers ausgeführt. Gehört dieses Programm oder Skript *root*, so läuft es mit Root-Rechten und allen daraus resultierenden Konsequenzen.

Setzen können Sie das SUID-Bit wieder auf zwei Arten:

Beispiel

Attributmodus

```
root@frodo:~# chmod u=rwxs /var/permissions_directory/permissions.sh
```

Beispiel

Oktalschreibweise

```
root@frodo:~# chmod 4777 /var/permissions_directory/permissions.sh
```

Zum Entfernen haben Sie auch wieder zwei Möglichkeiten:

Beispiel

Attributmodus

```
root@frodo:~# chmod u-s /var/permissions_directory/permissions.sh
```

Beispiel

Oktalschreibweise

```
root@frodo:~# chmod 0777 /var/permissions_directory/permissions.sh
```

Überlegen Sie immer mehr als zweimal, ob Sie das SUID-Bit benötigen. Es ist mit das Gefährlichste, was Sie auf Ihrem System tun können, gerade wenn es darum geht, ein Programm oder Skript als *root* laufen zu lassen.

8.3.3 SGID-Bit

Sie können ein Programm oder ein Skript nicht nur mit den Rechten des Besitzers laufen lassen (SUID-Bit), sondern auch mit den Rechten der zugewiesenen Gruppe. Dies bewerkstelligen Sie mit dem SGID-Bit (**S**et**G**roup**ID**-Bit). Möchten Sie beispielsweise, dass ein Programm oder Skript, das von jedem aus-

geführt werden darf und in eine Datei schreibt, nur von den Mitarbeitern gelesen werden darf, dann können Sie das SGID-Bit setzen.

Sobald Sie einen Webserver auf Ihrem System laufen haben, z. B. *Apache*, werden Sie früher oder später in die Verlegenheit kommen, dass dieser Webserver Daten speichern soll. Dies tut er natürlich nicht automatisch, sondern über ein Programm oder Skript (z. B. ein PHP-Skript). Auch der Webserver wird im System als Benutzer angelegt (z. Z. *www-data*).

Wird vom Webserver nun ein Programm oder Skript ausgeführt, läuft dies mit den Rechten des Benutzers *www-data*, und alle Dateien, die dieses Skript schreibt, gehören dem Benutzer sowie der Gruppe www-data. Möchten Ihre Mitarbeiter nun Änderungen an diesen Dateien vornehmen, indem sie das gleiche Skript oder Programm verwenden, haben sie dazu keine Rechte, weil das Skript nun z. B. mit den Benutzerrechten von *employee1* und der Gruppe *employees* im System läuft.

Die Lösung für dieses Problem wäre, dem Programm oder Skript die Gruppe *employees* zuzuweisen und darauf das SGID-Bit zu setzen. Dann wird das Programm oder Skript nach wie vor unter dem Namen des Webservers laufen, aber ebenfalls mit der Gruppe *employees*, egal von wem es ausgeführt wird. Dies hat zur Folge, dass auch die von dem Programm oder Skript erstellten Dateien dieser Gruppe angehören.

Zum Setzen des SGID-Bits können Sie wieder zwei Methoden verwenden:

Beispiel

Attributmodus

```
root@frodo:~# chmod g=rwxs /var/permissions_directory/permissions.sh
```

Beispiel

Oktalschreibweise

```
root@frodo:~# chmod 2777 /var/permissions_directory/permissions.sh
```

Zum Entfernen des SGID-Bits können Sie folgende Methoden verwenden:

Beispiel

Attributmodus

```
root@frodo:~# chmod g-s /var/permissions_directory/permissions.sh
```

Beispiel

Oktalschreibweise

```
root@frodo:~# chmod 0777 /var/permissions_directory/permissions.sh
```

Sie sollten sich eine Schreibweise, entweder mit Attributen oder die Oktalschreibweise, angewöhnen. Aus Platzgründen und Gründen der Übersichtlichkeit werde ich im weiteren Verlauf des Kurses nur noch die Oktalschreibweise verwenden.

8.4 Standardberechtigungen mit umask

Immer wenn Sie eine Datei oder ein Verzeichnis erstellen, wird für das erstellte Objekt eine Berechtigung vergeben. Nur, wo kommt diese Berechtigung her? Woher weiß Linux, welche Berechtigung auf ein Verzeichnis und welche auf eine Datei gesetzt wird?

Standardrechte, mit denen Dateien erstellt werden, können Sie mit dem Befehl `umask`, gefolgt von einer vierstelligen Ziffer, manipulieren. Diese Zahl liegt zwischen `0000` und `0777` und wird als die ***Umask*** bezeichnet. Sie »maskiert« die Rechte und nimmt von der vollen Maske die Rechte weg, die nicht gesetzt werden sollen.

Das Maximum für Dateien ist `666`, denn es können keine Dateien direkt mit Ausführungsbit angelegt werden. Hierzu müssen Sie manuell `chmod` bemühen. Verzeichnisse können das Ausführungsbit gesetzt haben (Wechsel in Verzeichnis).

Umask an einem Beispiel

Gesetzt den Fall, wir wollen, dass Dateien standardmäßig mit den Rechten `644` und Verzeichnisse mit den Rechten `755` angelegt werden, kann die Umask folgendermaßen ermittelt werden:

```
rw- rw- rw- Volle Dateimaske (666)
rwx rwx rwx Volle Maske für Verzeichnisse (777)
rw- r-- r-- Gewünschte Dateimaske (644)
rwx r-x r-x Gewünschte Verzeichnismaske (755)
--- -w- -w- Rechte, die nicht gesetzt werden (022) <-- Umask
```

Sie können sich die aktuelle *Umask* mit dem Befehl `umask` anzeigen lassen, indem Sie keinen Wert übergeben:

Beispiel

```
root@frodo:~# umask
0022
```

Zum Setzen der *Umask* genügt es, den Befehl `umask` mit der gewünschten Maske abzusetzen. Wollen Sie beispielsweise, dass Ihre Dateien als Standard die Rechte 640 und Ihre Verzeichnisse 750 erhalten, dann setzen Sie die *Umask* wie folgt:

Beispiel

```
root@frodo:~# umask 0027
```

Wenn Sie jetzt eine Datei erstellen, wird die neue Umask angewendet, und die Berechtigungen auf die Datei werden dementsprechend gesetzt:

```
root@frodo:~# touch test.sh

root@frodo:~# ls -l test.sh
-rw-r----- 1 root root 0 Jul 14 15:29 test.sh
```

Die so gesetzte *Umask* ist nur für die Dauer der Shellsitzung aktiv, danach wird wieder der Standardwert verwendet. Möchten Sie den Standardwert für einen bestimmten User verändern, so können Sie das über die Datei *~/.profile* tun.

Sobald eine Shell gestartet wird, geht Linux unter anderem die »profile-Dateien« durch: zuerst die Datei */etc/profile* und anschließend die Datei *~/.profile* im Homeverzeichnis des jeweiligen Users.

Möchten Sie die *Umask* global ändern, so wurde das früher über die Datei */etc/profile* getan. Inzwischen wurde das Setzen der globalen *Umask* aber *PAM* und der Datei */etc/login.defs* übertragen. Mehr zu *PAM* erfahren Sie in Lektion 16 »PAM«.

Und, hat es funktioniert? Wenn Sie beim Lesen bereits auf die Frage und deren Lösung gekommen sind, sind Sie auf dem besten Weg, ein Administrator zu werden.

Übung 18:

1. Bearbeiten Sie Ihre Konfiguration so, dass Dateien, die *customer2* erstellt, standardmäßig die Rechte `-rw-r----` erhalten und Verzeichnisse, die von *customer2* erstellt werden, nur für *customer2* und die Gruppe *customers* zugänglich sind.

2. Testen Sie Ihre Konfiguration, indem Sie sich als *customer2* anmelden und ein Verzeichnis sowie darin zwei Dateien anlegen.

3. Damit Sie sich nicht über ein zusätzliches Terminal als *customer2* anmelden müssen, können Sie den Befehl `su - customer2` im aktuellen Terminal verwenden und werden so zu *customer2*. Mit dem Befehl `exit` kehren Sie zu *root* zurück.

8.5 Testen Sie Ihr Wissen

1. Sie möchten für das Verzeichnis */var/backups/users* und alle darin enthaltenen Dateien und Verzeichnisse die Gruppe zu `admin` ändern. Welche der folgenden Befehle können Sie nutzen?

   Bitte ankreuzen:

   - ☐ `chmod -R :admin /var/backups/users`
   - ☐ `chown -R :admin /var/backups/users`
   - ☐ `chown -R admin /var/backups/users`
   - ☐ `chgrp -R :admin /var/backups/users`
   - ☐ `chgrp -R admin /var/backups/users`

2. Welche Aussagen über das Verzeichnis */var/employees* treffen zu?

   ```
   drwxrwxrwt 2 root root 4096 Jul 14 11:50 /var/employees
   ```

   Bitte ankreuzen:

   - ☐ Im Verzeichnis */var/employees* dürfen nur die Besitzer einer Datei diese ändern oder löschen.
   - ☐ Im Verzeichnis */var/employees* werden die Rechte beim Ausführen der Datei auf den Besitzer der Datei gesetzt.
   - ☐ Die Berechtigung `drwxrwxrwt` kann mit dem Kommando `chmod 4777 /var/employees` gesetzt werden.
   - ☐ Die Berechtigung `drwxrwxrwt` kann mit dem Kommando `chmod 1777 /var/employees` gesetzt werden.
   - ☐ Es ist das Sticky Bit gesetzt.
   - ☐ Es ist das SUID-Bit gesetzt.

3. Sie sind der User **johndoe**. Die Ausgabe des Befehls `ls -la` bringt folgendes Ergebnis:

   ```
   drwxrwxrwx        2  root    root        96  Jul    8  18:54 .
   drwxrwxrwx        5  root    root       784  Jul    8  18:54 ..
   -rw-rw-rw-        2  root    root         0  Jul    8  18:54 eins.txt
   -rw-r--r--        2  root    root         0  Jul    8  18:54 zwei.txt
   ```

   Sie setzen in dem Verzeichnis den Befehl `rm * -f` ab. Was passiert?

   Bitte ankreuzen:

   - ☐ Nur Datei **eins.txt** wird gelöscht.
   - ☐ Nur Datei **zwei.txt** wird gelöscht.
   - ☐ Beide Dateien (**eins.txt** und **zwei.txt**) werden gelöscht.
   - ☐ Keine der Dateien wird gelöscht.