5.3 Lektion 1

Zertifikat:	Linux Essentials
Version:	1.6
Thema:	5 Sicherheit und Dateiberechtigungen
Lernziel:	5.3 Dateiberechtigungen und Dateieigentum verwalten
Lektion:	1 von 1

Einführung

Linux als Mehrbenutzersystem muss nachvollziehen, wem welche Dateien gehören und ob ein Benutzer Aktionen mit bestimmten Dateien ausführen darf oder nicht. Nur so ist die Privatsphäre der Benutzer zu gewährleisten, die den Inhalt ihrer Dateien vertraulich behandelt wissen möchten. Gleichzeitig macht dies Zusammenarbeit erst möglich, indem bestimmte Dateien mehreren Benutzern zugänglich sind.

Dies geschieht durch ein dreistufiges Berechtigungssystem: Jede Datei auf der Festplatte gehört einem Benutzer und einer Benutzergruppe und hat drei Berechtigungsgruppen: eine für den Eigentümer, eine für die Gruppe, der die Datei gehört, und eine für alle anderen. In dieser Lektion lernen Sie, wie Sie die Berechtigungen für eine Datei abfragen und wie Sie sie manipulieren.

Informationen über Dateien und Verzeichnisse abfragen

Der Befehl `ls` dient dazu, den Inhalt eines beliebigen Verzeichnisses aufzulisten. In der Grundform erhalten Sie nur die Datei- und Verzeichnisnamen:

```
$ ls
Another_Directory  picture.jpg  text.txt
```

Aber es gibt viel ausführlichere Informationen für jede Datei, einschließlich Typ, Größe, Eigentümer und mehr. Um diese Informationen zu sehen, rufen Sie `ls` mit dem Parameter `-l` für eine Liste in “Langform” auf:

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Die Spalten in der obigen Ausgabe haben folgende Bedeutung:

- Die *erste* Spalte der Liste zeigt den Dateityp und die Zugriffsrechte an, zum Beispiel: `drwxrwxr-x`:
    - Das erste Zeichen, `d`, zeigt den Dateityp an.
    - Die nächsten drei Zeichen, `rwx`, zeigen die Rechte des *Besitzers* der Datei an, auch *user* oder `u` genannt.
    - Die nächsten drei Zeichen, `rwx`, zeigen die Rechte der *Gruppe* an, der die Datei gehört, auch als `g` bezeichnet.
    - Die letzten drei Zeichen, `r-x`, zeigen die Rechte für jeden anderen an, auch als *other* oder `o` bezeichnet.

- Die *zweite* Spalte gibt die Anzahl der Hardlinks an, die auf diese Datei zeigen. Für ein Verzeichnis bedeutet dies die Anzahl der Unterverzeichnisse plus einen Link auf sich selbst (`.`) und das übergeordnete Verzeichnis (`..`).
- Die *dritte* und *vierte* Spalte zeigen die Eigentümer bzw. Benutzer und Gruppe, die die Datei besitzen.
- Die *fünfte* Spalte zeigt die Dateigröße in Bytes an.
- Die *sechste* Spalte zeigt das genaue Datum und die genaue Zeit, den sogenannten *Zeitstempel*, wann die Datei zuletzt geändert wurde.
- Die *siebente* und letzte Spalte zeigt den Dateinamen.

Wenn Sie die Dateigrößen in einem “menschenlesbaren” Format sehen möchten, fügen Sie den Parameter `-h` zu `ls` hinzu. Dateien mit einer Größe von weniger als einem Kilobyte werden in Bytes angezeigt. Bei Dateien mit mehr als einem Kilobyte und weniger als einem Megabyte wird nach der Größe ein `K` hinzugefügt, das die Größe in Kilobyte anzeigt. Dasselbe gilt für Dateigrößen im Megabyte- (`M`) und Gigabyte-Bereich (`G`):

```
$ ls -lh
total 1,2G
drwxrwxr-x 2 carol carol 4,0K Dec 10 17:59 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
-rw------- 1 carol carol 528K Dec 10 10:43 picture.jpg
---xr-xr-x 1 carol carol   33 Dec 11 10:36 test.sh
-rwxr--r-- 1 carol carol 1,9K Dec 20 18:13 text.txt
-rw-rw-r-- 1 carol carol 2,6M Dec 11 22:14 Zipped.zip
```

Um nur Informationen über einen bestimmten Satz von Dateien anzuzeigen, fügen Sie die Namen dieser Dateien zu `ls` hinzu:

```
$ ls -lh HugeFile.zip test.sh
total 1,2G
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
---xr-xr-x 1 carol carol   33 Dec 11 10:36 test.sh
```

Was ist mit Verzeichnissen?

Wenn Sie versuchen, Informationen über ein Verzeichnis mit `ls -l` abzufragen, zeigt es Ihnen ein Liste des Verzeichnisinhalts:

```
$ ls -l Another_Directory/
total 0
-rw-r--r-- 1 carol carol 0 Dec 10 17:59 another_file.txt
```

Um dies zu vermeiden und Informationen über das Verzeichnis selbst abzufragen, fügen Sie den Parameter `-d` zu `ls` hinzu:

```
$ ls -l -d Another_Directory/
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory/
```

Versteckte Dateien sehen

Die Verzeichnisliste, die wir zuvor mit `ls -l` abgerufen haben, ist unvollständig:

```
$ ls -l
total 544
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Es gibt drei weitere Dateien in diesem Verzeichnis, aber sie sind versteckt. Unter Linux werden Dateien, deren Name mit einem Punkt (`.`) beginnt, automatisch versteckt. Um sie zu sehen, müssen wir den Parameter `-a` zu `ls` hinzufügen:

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol   4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol   4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol      0 Dec 10 16:01 .thisIsHidden
```

Die Datei `.thisIsHidden` wird nur dadurch versteckt, dass ihr Name mit `.` beginnt.

Die Verzeichnisse `.` und `..` sind Sonderfälle: `.` ist ein Zeiger auf das aktuelle Verzeichnis, während `..` ein Zeiger auf das übergeordnete Verzeichnis ist (das Verzeichnis, das das aktuelle Verzeichnis enthält). Unter Linux enthält jedes Verzeichnis mindestens diese beiden speziellen Verzeichnisse.

Tip Sie können mehrere Parameter für `ls` (und viele andere Linux-Befehle) kombinieren. `ls -l -a` kann zum Beispiel als `ls -la` geschrieben werden.

Dateitypen verstehen

Wir haben bereits erwähnt, dass der erste Buchstabe in jeder Ausgabe von `ls -l` den Typ der Datei beschreibt. Die drei häufigsten Dateitypen sind:

`-` (normale Datei)
: Eine Datei kann Daten jeglicher Art enthalten. Dateien können verändert, verschoben, kopiert und gelöscht werden.

`d` (Verzeichnis)
: Ein Verzeichnis enthält andere Dateien oder Verzeichnisse und hilft, das Dateisystem zu organisieren. Technisch gesehen sind Verzeichnisse eine besondere Art von Dateien.

`l` (Soft Link)
: Diese “Datei” ist ein Zeiger auf eine andere Datei oder ein anderes Verzeichnis im Dateisystem.

Neben diesen gibt es drei weitere Dateitypen, die Sie zumindest kennen sollten, die aber nicht Thema dieser Lektion sind:

`b` (Block Device)
: Diese Datei steht für ein virtuelles oder physisches Gerät, normalerweise Festplatten oder andere Arten von Speichergeräten. Zum Beispiel könnte die erste Festplatte im System durch `/dev/sda` dargestellt werden.

`c` (Character Device)
: Diese Datei steht für ein virtuelles oder physisches Gerät. Terminals (wie das Hauptterminal auf `/dev/ttyS0`) und serielle Schnittstellen sind gängige Beispiele für Character Devices.

`s` (Socket)

Sockets dienen als “Leitungen”, die Informationen zwischen zwei Programmen weiterleiten.

Warning Ändern Sie keine der Berechtigungen auf Block Devices, Character Devices oder Sockets, es sei denn, Sie wissen, was Sie tun. Andernfalls könnte Ihr System nicht mehr funktionieren!

Berechtigungen verstehen

In der Ausgabe von `ls -l` werden die Dateiberechtigungen direkt nach dem Dateityp als drei Gruppen von je drei Zeichen in der Reihenfolge `r`, `w` und `x` angezeigt. Beachten Sie, dass ein Strich `-` das Fehlen einer bestimmten Berechtigung anzeigt.

Berechtigungen auf Dateien

`r`

Steht für “read”, also “lesen”, und hat einen Oktalwert von `4` (wir werden in Kürze auf Oktale eingehen). Das bedeutet die Berechtigung, eine Datei zu öffnen und ihren Inhalt zu lesen.

`w`

Steht für “write”, also “schreiben”, und hat einen Oktalwert von `2`. Das bedeutet die Berechtigung, eine Datei zu bearbeiten oder zu löschen.

`x`

Steht für “execute”, also “ausführen”, und hat einen oktalen Wert von `1`. Das bedeutet, dass die Datei als ausführbare Datei oder als Skript ausgeführt werden kann.

So kann z.B. eine Datei mit den Rechten `rw-` zwar gelesen und geschrieben, aber nicht ausgeführt werden.
Berechtigungen auf Verzeichnisse

`r`

Steht für “read” und hat einen oktalen Wert von `4`. Das bedeutet die Berechtigung, den Inhalt des Verzeichnisses zu lesen, wie z.B. Dateinamen. Aber es bedeutet *nicht* die Berechtigung, die Dateien selbst zu lesen.

`w`

Steht für “write” und hat den oktalen Wert `2`. Das bedeutet die Berechtigung, Dateien in einem Verzeichnis zu erstellen oder zu löschen oder deren Namen, Rechte und Besitzer zu ändern. Wenn ein Benutzer die Schreibberechtigung für ein Verzeichnis hat, kann er die Berechtigungen jeder Datei in dem Verzeichnis ändern, auch wenn er keine Rechte auf die Datei hat oder die Datei einem anderen Benutzer gehört.

`x`

Steht für “execute” und hat einen oktalen Wert von `1`. Das bedeutet die Berechtigung, in ein Verzeichnis zu wechseln, aber nicht, seine Dateien aufzulisten (dafür ist die `r`-Berechtigung erforderlich).

Die letzte Erklärung mag verwirrend klingen. Stellen wir uns darum zum Beispiel vor, Sie haben ein Verzeichnis namens `Another_Directory` mit den folgenden Rechten:

```
$ ls -ld Another_Directory/
```

```
d--xr-xr-x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Nehmen Sie darüber hinaus an, dass Sie in diesem Verzeichnis ein Shell-Skript namens `hello.sh` mit den folgenden Rechten liegt:

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

Wenn Sie der Benutzer `carol` sind und versuchen, den Inhalt von `Another_Directory` aufzulisten, erhalten Sie eine Fehlermeldung, da Ihr Benutzer keine Leseberechtigung für dieses Verzeichnis hat:

```
$ ls -l Another_Directory/
ls: cannot open directory 'Another_Directory/': Permission denied
```

Der Benutzer `carol` hat jedoch Ausführungsrechte, d.h. er darf das Verzeichnis betreten und auf Dateien innerhalb des Verzeichnisses zugreifen, sofern er die richtigen Rechte *für die jeweilige Datei* hat. In diesem Beispiel hat der Benutzer volle Rechte für das Skript `hello.sh`, so dass er das Skript *ausführen* kann, auch wenn er den Inhalt des Verzeichnisses, in dem es sich befindet, *nicht* lesen kann. Es genügt der vollständige Dateiname.

```
$ sh Another_Directory/hello.sh
Hello LPI World!
```

Wie bereits erwähnt, werden die Rechte in der folgenden Reihenfolge angegeben: zuerst für den Eigentümer der Datei, dann für die besitzende Gruppe und dann für andere Benutzer. Immer wenn jemand versucht, eine Aktion an der Datei durchzuführen, werden die Rechte auf die gleiche Weise geprüft. Zuerst wird geprüft, ob der aktuelle Benutzer die Datei besitzt, und wenn dies zutrifft, wird nur der erste Satz von Rechten angewendet, andernfalls wird geprüft, ob der aktuelle Benutzer zu der Gruppe gehört, der die Datei gehört. In diesem Fall wird nur der zweite Satz von Rechten angewendet. In jedem anderen Fall wird der dritte Satz von Rechten angewendet. Das heißt, wenn der aktuelle Benutzer der Eigentümer der Datei ist, sind nur die Rechte des Eigentümers wirksam, auch wenn die Gruppen- oder sonstigen Rechte permissiver sind als die Rechte des Eigentümers.

Ändern von Dateiberechtigungen

Der Befehl `chmod` wird verwendet, um die Rechte für eine Datei zu ändern, und braucht mindestens zwei Parameter: der erste beschreibt, welche Rechte geändert werden sollen, und der zweite zeigt auf die Datei oder das Verzeichnis, an dem die Änderung vorgenommen wird. Die Rechte zum Ändern können jedoch auf zwei verschiedene Arten oder *Modes* beschrieben werden.

Der erste, der *symbolic Mode*, bietet eine differenzierte Steuerung, die es Ihnen erlaubt, eine einzelne Berechtigung hinzuzufügen oder zu widerrufen, ohne andere im Set zu ändern. Der zweite, der *numeric Mode*, ist leichter zu merken und schneller zu benutzen, wenn Sie alle Berechtigungswerte auf einmal setzen wollen.

Beide Modi führen zum gleichen Ergebnis. Die Befehle:

```
$ chmod ug+rw-x,o-rwx text.txt
```

und

```
$ chmod 660 text.txt
```

erzeugen also eine Datei mit denselben Rechten:

```
-rw-rw---- 1 carol carol  765 Dec 20 21:25 text.txt
```

Schauen wir uns die beiden Modi genauer an.

Symbolic Mode

Wenn Sie beschreiben, welche Berechtigungen im *symbolischen Modus* geändert werden sollen, geben die ersten Zeichen an, welche Berechtigungen Sie ändern werden: die für den Benutzer (`u`), für die Gruppe (`g`), für andere (`o`) und/oder für alle drei zusammen (`a`).

Dann müssen Sie dem Befehl sagen, was er tun soll: Sie können eine Berechtigung erteilen (`+`), eine Berechtigung widerrufen (`-`) oder eine Berechtigung auf einen bestimmten Wert setzen (`=`).

Zuletzt geben Sie an, welche Rechte Sie beeinflussen wollen: Lesen (`r`), Schreiben (`w`) oder Ausführen (`x`).

Stellen Sie sich zum Beispiel vor, wir haben eine Datei namens `text.txt` mit dem folgenden Berechtigungssatz:

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Wenn Sie den Mitgliedern der Gruppe, der die Datei gehört, Schreibrechte geben wollen, würden Sie den Parameter `g+w` verwenden. Es ist einfacher, wenn Sie es so betrachten: “Gruppe (`g`) gewähren (`+`) Schreibrechte (`w`) .” Der Befehl wäre also:

```
$ chmod g+w text.txt
```

Prüfen wir das Ergebnis mit `ls`:

```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Wenn Sie die Leserechte für den Besitzer derselben Datei entfernen wollen, denken Sie: “Benutzer (`u`) widerrufen (`-`) Leserechte (`r`).” Der Parameter ist also `u-r`:

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Was ist zu tun, wenn wir die Berechtigungen für jeden genau auf `rw-` setzen wollen? Dann stellen Sie sich vor: “Alle (`a`) setzen (`=`) lesen (`r`), schreiben (`w`) und nicht ausführen (`-`).” Also:

```
$ chmod a=rw- text.txt
$ ls -l text.txt
-rw-rw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

Natürlich ist es möglich, mehrere Berechtigungen gleichzeitig zu modifizieren. In diesem Fall trennen Sie sie mit einem Komma (`,`):

```
$ chmod u+rwx,g-x text.txt
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

Das obige Beispiel kann wie folgt gelesen werden: “Benutzer (`u`) gewähren (`+`) Lese-, Schreib- und Ausführungsrechte (`rwx`) und Gruppe (`g`) widerrufen (`-`) Ausführungsrecht (`x`).”

Wenn `chmod` auf einem Verzeichnis ausgeführt wird, ändert es nur die Rechte des Verzeichnisses. `chmod` hat einen rekursiven Modus, wenn Sie etwa die Rechte für alle Dateien innerhalb eines Verzeichnisses und seiner Unterverzeichnisse ändern wollen. Dazu fügen Sie den Parameter `-R` nach dem Befehlsnamen und vor den zu ändernden Rechten hinzu:

```
$ chmod -R u+rwx Another_Directory/
```

Dieser Befehl kann wie folgt gelesen werden: “Rekursiv (`-R`) Benutzer (`u`) gewähren (`+`) Lese-, Schreib- und Ausführungsrechte (`rwx`).”

Warning Seien Sie vorsichtig und denken Sie zweimal nach, bevor Sie den `-R`-Schalter benutzen, da es einfach ist, Berechtigungen auf Dateien und Verzeichnisse zu ändern, die Sie nicht ändern wollen, besonders auf Verzeichnisse mit einer großen Anzahl von Dateien und Unterverzeichnissen.

Numeric Mode

Im *numerischen Modus* werden die Berechtigungen auf andere Weise angegeben: als dreistelliger numerischer Wert in oktaler Notation (ein Basis-8-Zahlensystem).

Jede Berechtigung hat einen entsprechenden Wert, und diese werden in der folgenden Reihenfolge angegeben: Zuerst Lesen (`r`), entspricht `4`, dann Schreiben (`w`), entspricht `2`, und zuletzt Ausführen (`x`), entspricht `1`. Soll keine Berechtigung gelten, entspricht dies dem Wert `0`. Die Berechtigung `rwx` entspricht also `7` (`4+2+1`) und `r-x` entspricht `5` (`4+0+1`).

Die erste der drei Ziffern des Berechtigungssatzes repräsentiert die Rechte für den Benutzer (`u`), die zweite für die Gruppe (`g`) und die dritte für andere (`o`). Um die Rechte für eine Datei auf `rw-rw----` zu setzen, ist der oktale Wert `660` anzugeben:

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Abgesehen davon entspricht die Syntax im numerischen Modus der des symbolischen Modus: Der erste Parameter repräsentiert die Rechte, die Sie ändern möchten, und der zweite zeigt auf die Datei oder das Verzeichnis, in dem die Änderung vorgenommen wird.

Tip Wenn ein Berechtigungswert *ungerade* ist, ist die Datei sicher ausführbar!

Welche Syntax ist zu verwenden? Der numerische Modus wird empfohlen, wenn Sie die Berechtigungen auf einen bestimmten Wert ändern wollen, zum Beispiel `640` (`rw- r-- ---`).

Der *symbolische Modus* ist nützlicher, wenn Sie nur einen bestimmten Wert ändern wollen, unabhängig von den aktuellen Berechtigungen für die Datei. Zum Beispiel können Sie Ausführungsberechtigungen für den Benutzer hinzufügen, indem Sie einfach `chmod u+x script.sh` benutzen, ohne die aktuellen Berechtigungen für die Gruppe und andere berücksichtigen oder gar ändern zu müssen.

Ändern des Dateibesitzes

Der Befehl `chown` dient dazu, die Besitzverhältnisse einer Datei oder eines Verzeichnisses zu ändern. Die Syntax ist recht einfach:

```
chown username:groupname filename
```

Schauen wir uns zum Beispiel eine Datei namens `text.txt` an:

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

Der Benutzer, dem die Datei gehört, ist `carol`, und die Gruppe ist ebenfalls `carol`. Wir wollen nun die Datei an eine andere Gruppe, nämlich `students`, übertragen:

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

Beachten Sie, dass der Benutzer, der eine Datei besitzt, nicht der Gruppe angehören muss, der die Datei gehört. Im obigen Beispiel muss der Benutzer `carol` nicht Mitglied der Gruppe `students` sein, aber er muss Mitglied der Gruppe sein, um den Gruppenbesitz der Datei auf diese Gruppe zu übertragen.

Benutzer oder Gruppen können weggelassen werden, wenn Sie diese nicht ändern wollen. Um nur die Gruppe zu ändern, der eine Datei gehört, würden Sie also `chown :students text.txt` verwenden. Um nur den Benutzer zu ändern, wäre der Befehl chown `chown carol text.txt`. Alternativ könnten Sie den Befehl `chgrp students text.txt` verwenden, um nur die Gruppe zu ändern.

Wenn Sie nicht der Systemadministrator (root) sind, können Sie den Besitzer einer Datei, die einem anderen Benutzer oder einer Gruppe gehört, der Sie nicht angehören, nicht ändern. Wenn Sie versuchen, dies zu tun, erhalten Sie die Fehlermeldung `Operation not permitted`.

Gruppen abfragen

Bevor Sie die Besitzrechte einer Datei ändern, kann es nützlich sein zu wissen, welche Gruppen auf dem System existieren, welche Benutzer Mitglieder einer Gruppe sind und zu welchen Gruppen ein Benutzer gehört. Diese Aufgaben erledigen Sie mit den Befehlen `groups` und `groupmems`.

Um zu sehen, welche Gruppen auf Ihrem System existieren, geben Sie einfach `groups` ein:

```
$ groups
carol students cdrom sudo dip plugdev lpadmin sambashare
```

Und wenn Sie wissen wollen, zu welchen Gruppen ein Benutzer gehört, fügen Sie den Benutzernamen als Parameter hinzu:

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

Um den umgekehrten Weg zu gehen und anzuzeigen, welche Benutzer zu einer Gruppe gehören, benutzen Sie `groupmems`. Der Parameter `-g` gibt die Gruppe an, und `-l` listet alle ihre Mitglieder auf:

```
$ sudo groupmems -g cdrom -l
carol
```

Tip `groupmems` kann nur als root ausgeführt werden. Wenn Sie nicht als root angemeldet sind, fügen Sie vor dem Befehl `sudo` hinzu.

Spezielle Berechtigungen

Neben den Lese-, Schreib- und Ausführungsrechten für Benutzer, Gruppen und andere kann jede Datei drei weitere spezielle Berechtigungen haben, die die Funktionsweise eines Verzeichnisses oder die Ausführung eines Programms beeinflussen. Sie können entweder im symbolischen oder im numerischen Modus angegeben werden und sind im Folgenden beschrieben.

Sticky Bit

Das Sticky Bit, auch *Restricted Deletion Flag* genannt, hat den oktalen Wert `1` und wird im symbolischen Modus durch ein `t` in den Rechten für andere repräsentiert. Dies gilt nur für Verzeichnisse, und unter Linux verhindert es, dass Benutzer eine Datei in einem Verzeichnis entfernen oder umbenennen können, wenn sie diese Datei oder dieses Verzeichnis nicht besitzen.

Verzeichnisse mit gesetztem Sticky Bit zeigen ein `t`, das das `x` in den Berechtigungen für *andere* in der Ausgabe von `ls -l` ersetzt:

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

Im numerischen Modus werden die speziellen Berechtigungen in einer “vierstelligen Notation” angegeben, wobei die erste Ziffer die spezielle Berechtigung darstellt, die man setzt. Um zum Beispiel das Sticky Bit (Wert `1`) für das Verzeichnis `Another_Directory` mit den Berechtigungen `755` im numerischen Modus zu setzen, lautet der Befehl:

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Set GID

Set GID, auch SGID oder Set Group ID Bit genannt, hat den oktalen Wert `2` und wird im symbolischen Modus durch ein `s` auf den *Gruppen*-Berechtigungen repräsentiert. Dies kann auf ausführbare Dateien oder Verzeichnisse angewendet werden. Bei ausführbaren Dateien gewährt es dem Prozess, der aus der Ausführung der Datei resultiert, Zugriff auf die Berechtigungen der Gruppe, der die Datei gehört. Auf Verzeichnisse angewendet, erbt jede Datei oder jedes Verzeichnis, das darunter erstellt wird, die Gruppe vom übergeordneten Verzeichnis.

Dateien und Verzeichnisse mit SGID-Bit zeigen ein `s` anstelle des `x` in den Berechtigungen für die *Gruppe* bei der Ausgabe von `ls -l`:

```
$ ls -l test.sh
-rwxr-sr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Um SGID-Berechtigungen zu einer Datei im symbolischen Modus hinzuzufügen, lautet der Befehl:

```
$ chmod g+s test.sh
```

```
$ ls -l test.sh
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

Das folgende Beispiel wird Ihnen helfen, die Auswirkungen von SGID auf ein Verzeichnis besser zu verstehen. Angenommen es gibt ein Verzeichnis namens `Sample_Directory`, das dem Benutzer `carol` und der Gruppe `users` gehört und das folgende Berechtigungsstruktur hat:

```
$ ls -ldh Sample_Directory/
drwxr-xr-x 2 carol users 4,0K Jan 18 17:06 Sample_Directory/
```

Nun wechseln wir in dieses Verzeichnis und erstellen darin mit dem Befehl `touch` eine leere Datei:

```
$ cd Sample_Directory/
$ touch newfile
$ ls -lh newfile
-rw-r--r-- 1 carol carol 0 Jan 18 17:11 newfile
```

Wie wir sehen, gehört die Datei dem Benutzer `carol` und der Gruppe `carol`. Hätte das Verzeichnis jedoch die SGID-Berechtigung, wäre das Ergebnis ein anderes. Zunächst fügen wir das SGID-Bit zum `Sample_Directory` hinzu und überprüfen das Ergebnis:

```
$ sudo chmod g+s Sample_Directory/
$ ls -ldh Sample_Directory/
drwxr-sr-x 2 carol users 4,0K Jan 18 17:17 Sample_Directory/
```

Das `s` in den Gruppenberechtigungen zeigt an, dass das SGID-Bit gesetzt ist. Nun wechseln wir in dieses Verzeichnis und erstellen wieder eine leere Datei mit dem `touch`-Befehl:

```
$ cd Sample_Directory/
$ touch emptyfile
$ ls -lh emptyfile
 -rw-r--r-- 1 carol users 0 Jan 18 17:20 emptyfile
```

Wie wir sehen, ist die Gruppe, der die Datei gehört, `users`. Das SGID-Bit hat dafür gesorgt, dass die Datei den Gruppeneigentümer ihres Elternverzeichnisses erbt, also `users`.

Set UID

SUID oder Set User ID hat den oktalen Wert `4` und wird durch ein `s` in den *Benutzer*-Berechtigungen im symbolischen Modus dargestellt. Es gilt nur für Dateien und hat ähnliche Wirkung wie das SGID-Bit, nur dass der Prozess mit den Rechten des *Benutzers* ausgeführt wird, dem die Datei gehört. Dateien mit dem SUID-Bit zeigen in der Ausgabe von `ls -l` ein `s` anstelle des `x` in den Berechtigungen für den Benutzer:

```
$ ls -ld test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Sie können mehrere spezielle Berechtigungen in einem Parameter kombinieren, indem Sie sie addieren. Um also SGID (Wert `2`) und SUID (Wert `4`) im numerischen Modus für das Skript `test.sh` mit den Berechtigungen `755` zu setzen, geben Sie ein:

```
$ chmod 6755 test.sh
```

Das Ergebnis lautet:

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

Tip Wenn Ihr Terminal Farbe unterstützt, und das tun heute die meisten, können Sie schnell erkennen, ob diese speziellen Berechtigungen gesetzt sind, indem Sie einen Blick auf die Ausgabe von `ls -l` werfen. Für das Sticky Bit wird der Verzeichnisname möglicherweise in schwarzer Schrift mit blauem Hintergrund angezeigt. Dasselbe gilt für Dateien mit den Bits SGID (gelber Hintergrund) und SUID (roter Hintergrund). Die Farben können andere sein, abhängig von Linux-Distribution und Terminal-Einstellungen.

Geführte Übungen

1. Erstellen Sie ein Verzeichnis namens `emptydir` mit dem Befehl `mkdir emptydir`. Listen Sie nun mit `ls` die Berechtigungen für das Verzeichnis `emptydir` auf.
2. Erstellen Sie eine leere Datei namens `emptyfile` mit dem Befehl `touch emptyfile`. Nun fügen Sie mit `chmod` in symbolischer Schreibweise Ausführungsrechte für den Besitzer der Datei `emptyfile` hinzu und entfernen die Schreib- und Ausführungsrechte für alle anderen. Benutzen Sie dazu nur einen `chmod`-Befehl.
3. Wie lauten die Berechtigungen für eine Datei namens `text.txt`, nachdem Sie den Befehl `chmod 754 text.txt` benutzt haben?
4. Angenommen eine Datei namens `test.sh` ist ein Shell-Skript mit den folgenden Berechtigungen und Eigentumsverhältnissen:

   ```
   -rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
   ```

   - Was sind die Berechtigungen für den Eigentümer der Datei?
   - Wenn der Benutzer `john` dieses Skript ausführt, unter welchen Rechten wird es ausgeführt?
   - Wie lautet unter Verwendung der numerischen Notation die Syntax von `chmod`, um die spezielle Erlaubnis, die dieser Datei gewährt wurde, “aufzuheben”?
5. Betrachten Sie diese Datei:

   ```
   $ ls -l /dev/sdb1
   brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
   ```

   Welche Art von Datei ist `sdb1`? Und wer kann darauf schreiben?
6. Betrachten Sie die folgenden 4 Dateien:

   ```
   drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
   ----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
   -rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
   drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
   ```

   Notieren Sie die entsprechenden Berechtigungen für jede Datei und jedes Verzeichnis in numerischer 4-stelliger Schreibweise.

   `Another_Directory`

   `foo.bar`

`HugeFile.zip`

`Sample_Directory`

Offene Übungen

1. Versuchen Sie folgendes in einem Terminal: Erstellen Sie eine leere Datei namens `emptyfile` mit dem Befehl `touch emptyfile`. Nun "nullen" Sie die Rechte für die Datei mit `chmod 000 emptyfile` aus. Was wird passieren, wenn Sie die Rechte für `emptyfile` ändern, indem Sie nur *einen* Wert für `chmod` im numerischen Modus übergeben, zum Beispiel `chmod 4 emptyfile`? Was geschieht, wenn Sie zwei benutzen, wie etwa `chmod 44 emptyfile`? Was lässt sich daraus über die Art und Weise ableiten, wie `chmod` den numerischen Wert liest?
2. Können Sie eine Datei ausführen, für die Sie zwar die Berechtigung zum Ausführen, aber nicht zum Lesen haben (`--x`)? Warum oder warum nicht?
3. Beachten Sie die Berechtigungen für das temporäre Verzeichnis `/tmp` auf einem Linux-System:
   ```
   $ ls -l /tmp
   drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
   ```
   Benutzer, Gruppen und andere haben volle Rechte. Aber kann ein normaler Benutzer *alle* Dateien in diesem Verzeichnis löschen? Warum ist das so?
4. Eine Datei namens `test.sh` hat die folgenden Berechtigungen: `-rwsr-xr-x`, was bedeutet, dass das SUID-Bit gesetzt ist. Nun führen Sie die folgenden Befehle aus:
   ```
   $ chmod u-x test.sh
   $ ls -l test.sh
   -rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
   ```
   Was haben wir gemacht? Was bedeutet das große `S`?
5. Wie würden Sie ein Verzeichnis namens `Box` erstellen, in dem alle Dateien automatisch der Gruppe `users` gehören und nur von dem Benutzer gelöscht werden können, der sie erstellt hat?

Zusammenfassung

Als Mehrbenutzersystem braucht Linux eine Möglichkeit zu verfolgen, wem eine Datei gehört und wer auf sie zugreifen kann. Dies geschieht durch ein dreistufiges Berechtigungssystem, das Sie in dieser Lektion kennengelernt haben.

In dieser Lektion haben Sie erfahren, wie Sie mit `ls` Informationen über Dateiberechtigungen erhalten, wie Sie mit `chmod` kontrollieren oder ändern, wer eine Datei erstellen, löschen oder modifizieren darf (sowohl in *numerischer* als auch in *symbolischer* Schreibweise) und wie Sie mit `chown` und `chgrp` den Eigentümer von Dateien ändern können.

Die folgenden Befehle wurden in dieser Lektion behandelt:

`ls`
: Dateien auflisten, optional mit Details wie Berechtigungen.

`chmod`
: Ändern der Berechtigungen einer Datei oder eines Verzeichnisses.

`chown`
: Ändern des besitzenden Benutzers und/oder der Gruppe einer Datei oder eines Verzeichnisses.

`chgrp`
: Ändern der besitzenden Gruppe einer Datei oder eines Verzeichnisses.