Rechte im Dateisystem mehr als nur r,w,x

Erfahren Sie, wie Sie einfacher Verzeichnisstrukturen für Mitarbeiter bereitstellen und diese mit den benötigten Rechten und ACLs versehen können.

Stefan Kania 08. September 2015

Jeder der schon mal mit Linux auf der Kommandozeile gearbeitet hat und dort administrativ tätig war, kennt diese drei Buchstaben: "rwx" und das pro Eintrag im Dateisystem dreimal. Die Dateisystemberechtigungen! Aber was verbirgt sich noch hinter dem Begriff "Dateisystemberechtigungen"? Weit mehr als nur die drei Berechtigungen "read", "write" und "execute". Mit diesem Artikel werde ich weitere Facetten der Dateisystemberechtigungen wie "special Bits", "Access Control Lists" und "erweiterte Attribute" ansprechen.

Immer wieder höre ich in den Grundlagenseminaren zu Linux von Teilnehmern, dass man mit den drei einfachen Rechten "read", "write" und "execute" nicht viel anfangen

kann. Dass komplexe Abbildungen von Zugriffsrechten unter Windows viel besser seien. Wo reicht es denn heute noch aus, dass man nur einem Benutzer und einer Gruppe Rechte an einem Verzeichnis oder einer Datei geben kann? Aber nicht nur Einsteiger, auch so mancher Admin stößt immer wieder an seine Grenzen.

Ich will in diesem Artikel nicht nur erklären, wie die Rechte funktionieren, was man mit den Rechten alles machen kann, sondern ich will auch praktische Tipps geben, die vielleicht das Leben mit den Dateisystemrechten etwas angenehmer machen können. Neben den bekannten Dateisystemrechten gehören auch noch die "special Bits", die "Access Contol Lists (ACL)" und die erweiterten Attribute in ein gut geplantes Berechtigungskonzept. Auch diese Themen werde ich in diesem Artikel ansprechen.

Die altbekannten Buchstaben "r","w","x"

Im Schnelldurchlauf will ich hier noch mal die Rechte "read", "write" und "execute" erklären, wobei ich dabei auch die unterschiedlichen Bedeutungen der Rechte an Dateien und Verzeichnissen erläutern werde:

- **read**
  Das "read"-Recht an einer Datei bedeutet, die Datei kann zum Lesen geöffnet werden. Der Anwender der dieses Recht hat, kann den Inhalt lesen, aber nicht verändern. Das "read"-Recht an einem Verzeichnis erlaubt es, dass ein Anwender sich den Inhalt des Verzeichnisses mit "ls" anzeigen lassen kann.
- **write**
  Das "write"-Recht an einer Datei erlaubt es dem Anwender, den Inhalt einer Datei zu verändern. Er hat dadurch nicht das Recht, den Dateinamen zu ändern oder gar eine Datei zu löschen. Erst wenn ein Anwender das "write"-Recht an einem Verzeichnis hat, kann er die Einträge im Verzeichnis löschen, umbenennen und neue Einträge erstellen. Die Vergabe des "write"-Rechts an einem Verzeichnis sollte daher immer gut überlegt sein. Das Recht erlaubt es einem Anwender, alle Einträge in dem Verzeichnis zu löschen, auch wenn er an den Einträgen selbst keine Rechte hat. In allen heutigen Unix/Linux-Systemen muss der Anwender aber alle drei Rechte besitzen, um Einträge löschen, umbenennen oder erstellen zu können.
- **execute**
  Das "execute"-Recht erlaubt es einem Anwender, eine Datei auszuführen. Auf Binärdateien und Shell-Skripte muss immer das "execute"-Recht gesetzt sein, damit ein Anwender das Programm oder Shell-Skript ausführen kann. Hat ein Anwender das "execute"-Recht an einem Verzeichnis, kann er mit dem Kommando "cd" in das Verzeichnis wechseln.

user, group, other

Neben den Rechten selbst gibt es dann noch die Zuordnung der Rechte. Es gibt drei Berechtigungszuordnungen: Da wäre als erstes der Besitzer einer Datei (user), dann die besitzende Gruppe (group) und abschließend noch der Rest der Welt (other). Jeder dieser Berechtigungszuordnungen können die Rechte "read", "write" und "execute" zugeordnet werden. Ein Anwender kann aber immer nur die Rechte über eine der Berechtigungszuordnungen erhalten. Er ist also entweder der Besitzer einer Datei, dann erhält er die Rechte von "user", oder er ist Mitglied der besitzenden Gruppe, dann erhält er die Rechte von "group". Wenn er weder Besitzer noch Mitglied der besitzenden Gruppe ist, erhält er immer die Rechte von "other". Sie sehen schon, Rechte über "other" zu vergeben ist keine gute Lösung, da Sie den Zugriff auf Dateisystemeinträge nicht wirklich steuern können.

Immer alles Oktal

Systeme können nichts mit Buchstaben wie "r", "w" oder "x" anfangen, Systeme benötigen Zahlen und die am besten im Binärformat. Deshalb werden die Berechtigungen intern in Binärwerten mit drei Stellen abgebildet. Daraus entstehen dann die Oktalwerte für die Rechte wie Sie sie in der Tabelle sehen können. Egal ob für "user", "group" oder "other" – die Rechte haben dabei die folgenden Wertigkeiten:

```
Recht    Binärwert    Oktalwert
-------- ------------ ----------------
read      2^2          4
write     2^1          2
execute   2^0          1
```

Werden also alle Rechte an eine Berechtigungszuordnung vergeben, ergibt das einen Oktalwert von "7". Maximal also "777".

Woher kommen die Rechte im Dateisystem?

Wenn Sie eine Datei oder ein Verzeichnis anlegen, haben diese Einträge bereits Berechtigungen, aber wo kommen diese Berechtigungen her? Eine Vererbung wie Sie sie von Windows her kennen gibt es unter Linux nicht. Hier ist die "umask" für die Vergabe der Berechtigungen eines neuen Eintrags im Dateisystem verantwortlich. Sie können sich die Umask mit dem gleichnamigen Kommando anzeigen lassen. Hier ein Beispiel:

```
stefan@stefan:~% umask
022
```

Die drei Stellen der Umask stehen hier für eine Berechtigungszuordnungen. Die erste Stelle für "user", die zweite für "group" und die dritte für "other". Die Umask zeigt an, welche Rechte beim Anlegen eines neuen Eintrags im Dateisystem NICHT vergeben werden. Der Besitzer erhält immer alle Rechte, die besitzenden Gruppe alles außer dem Schreibrecht, genau wie der Rest der Welt. Das bedeutet in der Standardeinstellung kann der Rest der Welt immer in alle Verzeichnisse

wechseln und sich den Inhalt aller Dateien anzeigen lassen. Jeder Anwender kann über die Kommandozeile die Einstellung der Umask mit dem Kommando "umask <wert>" selbst anpassen. Später in diesem Artikel werde ich noch auf die Planung eines Berechtigungskonzeptes eingehen, dabei werde ich zeigen, wie man die Umask auch systemweit setzen können. Doch sehen wir uns einmal je einen neuen Eintrag für eine Datei und ein Verzeichnis an und vergleichen dieses mit der gesetzten Umask:

```
stefan@stefan:~% ls -l
insgesamt 4
-rw-r--r-- 1 stefan users    0 Aug  4 11:34 datei1
drwxr-xr-x 2 stefan users 4096 Aug  4 11:34 verzeichnis1
```

Hier sieht man, dass die Berechtigungen am Verzeichnis mit der Umask übereinstimmen. Der Gruppe und dem Rest der Welt wurde das Schreibrecht nicht vergeben. Der Besitzer hat alle Rechte. Aber was ist mit der Datei? Da fehlt bei allen drei Berechtigungszuordnungen das Execute-Recht. Das ist auch korrekt so! Denn das Betriebssystem überprüft beim Anlegen einer neuen Datei, ob es überhaupt Sinn macht, das Execute-Rechte an der Datei zu setzen. Bei allen nicht-binär-Dateien macht das Setzen des Execute-Rechts auch keinen Sinn, also setzt das System das Recht auch nicht. Nur wenn Sie einen Quellcode kompilieren und dabei eine ausführbare Datei entsteht, dann wird auch das Execute-Recht entsprechend der Umask gesetzt.

Wie werden Rechte gesetzt?

Nach dem Sie jetzt eine Einführung zu den Rechten erhalten haben, will ich jetzt erklären, wie die Rechte gesetzt werden und wer alles Rechte setzen kann. Auch das Ändern der besitzenden Gruppe und des Besitzer will ich in diesem Abschnitt erklären.

Die Rechte an einem Eintrag können immer vom "root" und dem Besitzer einer Datei geändert werden. Zum Ändern der Rechte wird das Kommando "chmod" verwendet. Das Kommando "chmod" kann dabei auf zwei verschiedene Arten angewendet werden: Einmal gibt es die relative Vergabe der Berechtigungen, bei der immer die derzeitige Berechtigung geändert wird. Dann gibt es noch die absolute Vergabe der Berechtigungen, bei der der komplette Satz an Berechtigungen für alle Berechtigungszuordnungen neu erstellt wird.

Relative Vergabe der Rechte im Dateisystem

Bei der relativen Vergabe der Berechtigungen können Sie jedes einzelne Recht für sich vergeben. Hier sehen Sie einige Beispiele:

```
stefan@stefan:~% chmod u+x datei1
stefan@stefan:~% ls -l datei1
-rwxr--r-- 1 stefan users 0 Aug  4 11:34 datei1

stefan@stefan:~% chmod g-r,o-r datei1
stefan@stefan:~% ls -l datei1
-rwx------ 1 stefan users 0 Aug  4 11:34 datei1
```

```
stefan@stefan:~% chmod a+r datei1
stefan@stefan:~% ls -l datei1
-rwxr--r-- 1 stefan users 0 Aug  4 11:34 datei1

stefan@stefan:~% chmod a+rx datei1
stefan@stefan:~% ls -l datei1
-rwxr-xr-x 1 stefan users 0 Aug  4 11:34 datei1
```

Wie Sie an den Beispielen sehen, können Sie jedes Recht einzeln setzen. Wenn Sie ein Recht zum Beispiel mit "chmod u+x datei1" ändern wollen, aber der Besitzer bereits das Execute-Recht an der Datei hat, ändert sich nichts. Auch sehen Sie in den Beispielen, dass Sie mit der Option "a+r" oder "a-r" ein oder mehrere Rechte für alle Berechtigungszuordnungen gleichzeitig ändern können.

Absolute Vergabe der Rechte im Dateisystem

Dabei gehen Sie ganz anders vor. Sie überlegen sich, welche Rechte Sie für alle drei Berechtigungszuordnungen vergeben wollen, diese rechnen Sie dann in den entsprechenden dreistelligen Oktalwert um und vergeben dann die Rechte für den Eintrag komplett neu. Als Beispiel:

```
stefan@stefan:~% ls -l datei1
-rwxr-xr-x 1 stefan users 0 Aug  4 11:34 datei1

stefan@stefan:~% chmod 600 datei1
stefan@stefan:~% ls -l datei1
-rw------- 1 stefan users 0 Aug  4 11:34 datei
```

Hier spielt es keine Rolle, welche Rechte vorher auf dem Eintrag gesetzt waren, alle Rechte werden überschrieben.

Ändern der besitzenden Gruppe

Die besitzende Gruppe kann sowohl vom "root" als auch vom Besitzer eines Eintrags geändert werden. Wobei es für den Besitzer eine Einschränkung gibt: Er kann einen Eintrag des Dateisystems nur an Gruppen übergeben, in denen er auch Mitglied ist. Für die Änderung der besitzenden Gruppe verwenden Sie das Kommando "chgrp". Das Beispiel zeigt, wie der Gruppenbesitz geändert wird:

```
stefan@stefan:~% chgrp cdrom datei1
stefan@stefan:~% ls -l datei1
-rw------- 1 stefan cdrom 0 Aug  4 11:34 datei1
```

Ändern des Besitzers

Den Besitzer eines Eintrages im Dateisystem - gleich ob Datei oder Verzeichnis - kann nur der "root" ändern. Mit dem Kommando "chown" kann der "root" nicht nur den Besitzer ändern, sondern auch gleichzeitig die besitzenden Gruppe. Auch hierfür einige Beispiele:

```
root@stefan:~# chown stka datei1
root@stefan:~# ls -l datei1
-rw------- 1 stka cdrom 0 Aug  4 11:34 datei1

root@stefan:~# chown stefan:users datei1
root@stefan:~# ls -l datei1
-rw------- 1 stefan users 0 Aug  4 11:34 datei1
```

Wie Sie sehen, wurde dieser Vorgang als Benutzer "root" durchgeführt.

Special Bits

Bis zu diesem Punkt ist das Thema für viele von Ihnen mehr oder weniger bekannt. Jetzt kommt die erste Erweiterung der Berechtigungen. In den letzten Abschnitten habe ich immer von drei Oktalgruppen für die Berechtigungen gesprochen. Aber es gibt vier dieser Gruppen. Vor den eigentlichen Dateisystemberechtigungen steht eine vierte Gruppe bestehend aus drei Bits, die zusätzliche Rechte geben oder auch Rechte nehmen kann. Die folgende Tabelle zeigt eine Übersicht der Bits:

```
Rechte      Binärwerte  Oktalwert
-----------------------------------------
SUID         2^2        4
SGID         2^1        2
Sticky Bit    2^0        1
```

Das SUID-Bit

Werfen wir doch einmal einen Blick auf die Rechte die an der Datei "/etc/shadow" vergeben sind:

```
stefan@stefan:~% ls -l /etc/shadow
-rw-r----- 1 root shadow 1360 Mai 27 08:52 /etc/shadow
```

Da sehen Sie, dass der "root" Lese- und Schreibrecht hat und die Gruppe "shadow" nur das Leserecht. In dieser Datei befinden sich die Passwortinformationen aller Benutzer des lokalen Systems. Nur aufgrund der Rechte an der Datei sieht es so aus, als hätte ein "normaler" Benutzer keine Rechte an dieser Datei. In dieser Datei befindet sich aber das Passwort eines jeden Benutzers. Ein Benutzer kann aber sein Passwort mit dem Kommando "passwd" ändern und greift damit schreibend auf die Datei zu. Wie kann das sein? Schauen wir deshalb mal auf die Berechtigungen des Programms "/usr/bin/passwd":

```
stefan@stefan:~% ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 47032 Jul 15 21:29 /usr/bin/passwd
```

Da fällt auf, dass an der Stelle, wo sonst beim Besitzer ein "x" steht, jetzt ein "s" steht. Dieses kleine "s" zeigt an, dass das SUID-Bit gesetzt ist. Was passiert jetzt, wenn ein "normaler" Benutzer das Programm "passwd" aufruft um sein Passwort zu ändern? Das System prüft, ob der Benutzer das Recht hat, das Programm zu starten. Da der Benutzer über "other" die Rechte "r-x" besitzt, kann der Benutzer das Programm aufrufen. Jetzt prüft das System zusätzlich, ob das SUID-Bit gesetzt ist. Ist das der Fall, wie bei dem Programm "passwd", erhält der Benutzer ein ZUSÄTZLICHE UID, nämlich die UID des Besitzers des Programms. Damit hat der Benutzer für die Laufzeit des Programms zwei UIDs, seine
eigene und die des "root". Da der "root" Schreibrechte an der Datei "/etc/passwd" hat, kann der Benutzer jetzt sein Passwort ändern. Programme, bei denen das SUID-Bit gesetzt ist, lassen sich nicht im Hintergrund starten. Das wäre auch fatal, da dann der Benutzer im Vordergrund "root"-Rechte hätte.

Das Setzen des SUID-Bits ist nur sinnvoll auf Binärdateien. Das Recht können Sie mit dem Kommando "chmod u+s <Programm>" setzen und entfernen mit "chmod u-s <Programm>". Damit das SUID-Bit genutzt werden kann, muss auch immer für den
Besitzer das "x"-Bit gesetzt sein. Da nach dem Setzen des SUID-Bits an der Stelle des "x" jetzt immer ein "s" steht, können Sie das gesetzte "x"-Bit daran erkennen, dass es sich bei dem "s" um ein kleines "s" handelt. Würde an der Stelle ein großes "S" stehen, wäre das "x"-Bit nicht gesetzt.

Das SGID-Bit

Mit dem SGID-Bit können Sie die Verwaltung der Rechtestruktur in Ihrem Dateisystem beeinflussen. Normalerweise gibt es bei Linux keine Vererbung der Berechtigungen im Dateisystem, aber durch den Einsatz des SGID-Bits können Sie das in einem bestimmten Rahmen ändern. Wenn Sie an einem Verzeichnis das SGID-Bit setzen, wird ab diesem Zeitpunkt jeder neue Eintrag unterhalb des Verzeichnisses immer der Gruppe gehören, die in diesem Verzeichnis als besitzende Gruppe eingetragen ist. Das SGID-Bit vererbt sich auch auf alle neu erstellten Unterverzeichnisse, die nach dem Setzen des SGID-Bits erzeugt werden. Wenn Sie auf Ihrer Verzeichnisstruktur das SGID-Bit an den einzelnen Abteilungsverzeichnissen setzen, gehören anschließend alle Einträge der entsprechenden Gruppe, ohne dass ein Benutzer seine Standardgruppe ändern müsste. Zusammen mit einer angepassten Umask können Sie dann bestimmte Verzeichnisse gezielt einer Gruppe zuordnen und die Rechte bestimmen. Mehr dazu folgt später im praktischen Teil dieses Artikels.

Der Einsatz des SGID-Bits ist nur sinnvoll, wenn es auf Verzeichnisse gesetzt wird. Es wird mit dem Kommando "chmod g+s <Verzeichnis>" gesetzt. Hier ein Beispiel:

```
stefan@stefan:~% chmod g+s verzeichnis1
stefan@stefan:~% ls -ld verzeichnis1
drwxr-sr-x 2 stefan users 4096 Aug  4 11:34 verzeichnis1
stefan@stefan:~% chgrp cdrom verzeichnis1
stefan@stefan:ueb~% cd verzeichnis1
stefan@stefan:~/verzeichnis1% mkdir verzeichnis1a
```

```
stefan@stefan:~/verzeichnis1% ls -ld verzeichnis1a
drwxr-sr-x 2 stefan cdrom 4096 Aug  4 17:31 verzeichnis1a
```

Hier sehen Sie, dass das neue Verzeichnis der neu zugeordneten Gruppe "cdrom" gehört und das SGID-Bit am Verzeichnis "verzeichnis1" gesetzt wurde. Auch das neue Unterverzeichnis "verzeichnis1/verzeichnis1a" hat das SGID-Bit gesetzt.

Das Sticky-Bit

Der Name dieses Bits stammt von seiner ursprünglichen Bedeutung. Wurde dieses Bit auf ein ausführbares Programm vom "root" gesetzt, dann blieb das Programm nach Beendigung im Arbeitsspeicher "kleben". Beim nächsten Aufruf wurde das Programm dann direkt aus dem Arbeitsspeicher gestartet. Dadurch wurde der Startvorgang des Programms beschleunigt. Linux nutzt das Sticky-Bit aber anders. Wenn Sie das Sticky-Bit an einem Verzeichnis setzen, kann nur noch der Besitzer einer Datei in dem Verzeichnis diese auch löschen. Auch wenn ein anderer Benutzer am Verzeichnis die Rechte "r,w,x" besitzt, ist er nicht berechtigt eine Datei zu löschen. Das Sticky-Bit macht nur Sinn, wenn es auf Verzeichnissen gesetzt wird. Das Sticky-Bit setzen Sie mit dem Kommando "chmod o+t <Verzeichnis>". Ein Beispiel:

```
stefan@stefan:~% mkdir verzeichnis2
stefan@stefan:~% chmod g+w,o+t verzeichnis2
stefan@stefan:~% ls -ld verzeichnis2
drwxrwxr-t 2 stefan users 4096 Aug  4 18:02 verzeichnis2

stefan@stefan:~% cd verzeichnis2
stefan@stefan:~/verzeichnis2% touch datei2
stefan@stefan:~/verzeichnis2% ls -l datei2
-rw-r--r-- 1 stefan users 0 Aug  4 18:04 datei2

stefan@stefan:berechtigungen-ia/verzeichnis2% su stka
Passwort:
stka@stefan:/home/stefan/verzeichnis2% rm datei2
rm: Normale leere Datei (schreibgeschützt) »datei2“ entfernen? y
rm: das Entfernen von »datei2“ ist nicht möglich: Vorgang nicht zulässig
```

Sie sehen hier, obwohl der Benutzer "stka" alle Rechte am Verzeichnis "verzeichnis2" über die Gruppe "users" (in der er Mitglied ist) erhält, kann er die Datei in dem Verzeichnis nicht löschen. Auch die Meldung ist eine ganz andere, als wenn ihm das Recht fehlen würde. Hier wird jetzt auf Grund des Sticky-Bits der Vorgang untersagt. Das Sticky-Bit ist ein sehr gutes Mittel, um in Verzeichnissen auf die mehrere Benutzer Zugriff haben, ein versehentliches Löschen von Dateien durch Nichtbesitzer zu verhindern. Das Bit schützt aber nicht vor der Veränderung des Inhalts von Dateien.

Die erweiterten POSIX-ACLs

Aber was tun Sie, wenn Sie an einem Verzeichnis unterschiedliche Rechte für unterschiedliche Gruppen vergeben wollen? Diese Aufgabe können Sie nicht mehr mit den einfachen Dateisystemrechten lösen. Dafür benötigen Sie die Access Control Lists (ACL). Hier geht es darum, die Dateisystemrechte zu erweitern. Mithilfe der ACL ist es möglich, dass mehrere Gruppen oder Benutzer an einer Datei oder einem Verzeichnis unterschiedliche Rechte erhalten. Damit Sie die ACLs nutzen können, müssen Sie als Erstes das Dateisystem für die ACL-Unterstützung anpassen. Alle gängigen Dateisysteme wie ext2, ext3, ext4, reiserfs und xfs unterstützen ACLs. Aber die ACLs müssen eventuell beim Mounten des Dateisystems als Option mit angegeben werden. Sie können die entsprechende Option natürlich auch direkt in dem Eintrag der "/etc/fstab" mit angeben, so werden die ACLs auch bei einem Systemstart sofort wieder aktiviert.

Warum nur "eventuell"? Weil bei vielen aktuellen Distributionen die Dateisysteme schon so kompiliert wurden, dass sie ACLs unterstützen. Wenn Sie die in diesem Abschnitt besprochenen Beispiele auf Ihrem System nachvollziehen können, ohne die Datei "/etc/fstab" anpassen zu müssen, dann unterstützt Ihr Dateisystem die ACLs ohne dass Sie weitere Optionen angeben müssen. Sollte das nicht der Fall sein, müssen Sie die Einträge in Ihrer "/etc/fstab" wie folgt anpassen:

```
/dev/hdb1       /daten              ext4    errors=remount-ro,acl 0       0
```

Hier wurde für eine Datenpartition die Option "acl" hinzugefügt. Nach einem "remount" mit dem Kommando "mount -o remount,rw /daten" stehen dann auf dieser Partition die ACLs zur Verfügung. Bei den ACLs wird zwischen den einfache ACLs und den default ACLs unterschieden. Zur Verwaltung der ACLs gibt es die Kommandos "setfacl" zum Setzen der ACLs und "getfacl" zum Auslesen der ACLs. Wenn diese Programme auf Ihrem System nicht vorhanden sind, müssen Sie das Paket "acl" nachinstallieren.

Die einfachen ACLs

Einfache ACLs gelten nur für das Verzeichnis oder die Datei, auf die sie angewendet wurden. Die einfachen ACLs werden nicht vererbt. Das folgende Listing zeigt, wie eine ACL auf ein Verzeichnis gesetzt wird:

```
stefan@stefan:~% mkdir verzeichnis3
stefan@stefan:~% ls -ld verzeichnis3
drwxr-xr-x 2 stefan users 4096 Aug  4 18:51 verzeichnis3

stefan@stefan:~% setfacl -m g:cdrom:rx verzeichnis3
stefan@stefan:~% ls -ld verzeichnis3
drwxr-xr-x+ 2 stefan users 4096 Aug  4 18:51 verzeichnis3

stefan@stefan:~% getfacl verzeichnis3
# file: verzeichnis3
# owner: stefan
# group: users
user::rwx
group::r-x
```

```
group:cdrom:r-x
mask::r-x
other::r-x
```

Im ersten Teil des Beispiels wird ein neues Verzeichnis angelegt und die Rechte aufgelistet. Im zweiten Teil wird dann mit dem Kommando "setfacl -m g:cdrom:rx verzeichnis3" ein ACL gesetzt. Die einzelnen Parameter haben dabei die folgende Bedeutung:

- **-m**
  Mit dieser Option "-m" wird das Kommando "setfacl" angewiesen, die ACL eines Eintrages zu modifizieren.
- **- g:cdrom:rx verzeichnis3**
  Eine Gruppe (bestimmt durch den Parameter "g") "cdrom" erhält die Rechte "rx" am "verzeichnis3"

Die Reihenfolge der Parameter muss dabei eingehalten werden.

Im dritten Teil werden die Rechte des Verzeichnisses "verzeichnis3" nochmal aufgelistet. Hier sehen Sie, dass nach den Rechten jetzt ein "+" zu sehen ist. Dieses "+" weist darauf hin, dass an dem Eintrag im Dateisystem ACLs gesetzt sind. Dann wird die ACL des Verzeichnisses "verzeichnis3" aufgelistet. Hier sehen Sie jetzt, dass neben der Gruppe "users" die Gruppe "cdrom" Rechte erhalten hat.

ACHTUNG: Wenn Sie auf einem Eintrag im Dateisystem – egal ob bei einem Verzeichnis oder einer Datei – eine ACL gesetzt haben, dürfen Sie von dem Moment an keine Berechtigungen mehr mit dem Kommando "chmod" ändern. Denn dann ändern Sie nicht mehr die Rechte sondern nur eine ACL-Maske. Diese Maske kann nur Rechte ausfiltern. Sprich: ein Recht, das dort nicht gesetzt ist, kann auch keine Gruppe oder Benutzer erhalten.

Um das Verhalten zu verdeutlichen, folgt hier ein Beispiel:

```
stefan@stefan:~% chmod g-r verzeichnis3
stefan@stefan:~% getfacl verzeichnis3
# file: verzeichnis3
# owner: stefan
# group: users
user::rwx
group::r-x                      #effective:--x
group:cdrom:r-x                 #effective:--x
mask::--x
other::r-x
```

Hier sehen Sie, dass die Gruppen nur noch das effektive Recht "x" besitzen, durch die Maske wird das "r"-Recht ausgefiltert. Wenn das Recht mit "chmod" wieder zurückgesetzt wird, wird auch die Maske wieder zurückgesetzt. Was passiert aber, wenn die besitzende Gruppe (so wie im Beispiel) nur die Rechte "r-x" besitzt und Sie für eine andere Gruppe die Rechte "rwx" setzen wollen? Denn in der Standardberechtigung fehlt ja das "w"-Recht. Da es sich bei den Berechtigungen aber um einen Filter handelt, müsste das "w"-Recht auch für eine andere Gruppe nicht wirksam sein. Aber

das ist nicht so, wie das folgende Listing zeigt:

```
stefan@stefan:~% setfacl -m g:cdrom:rwx verzeichnis3
stefan@stefan:~% getfacl verzeichnis3
# file: verzeichnis3
# owner: stefan
# group: users
user::rwx
group::r-x
group:cdrom:rwx
mask::rwx
other::r-x
```

Obwohl die besitzende Gruppe nur die Rechte "r-x" besitzt, hat die Gruppe "cdrom" alle Rechte am Verzeichnis. Erst eine nachträgliche Veränderung der Berechtigungsliste verändert die Maske. Selbstverständlich können Sie auch einem einzelnen Benutzer Rechte über ACLs vergeben. Das Listing zeigt diesen Vorgang:

```
stefan@stefan:~% setfacl -m u:stka:rwx verzeichnis3
stefan@stefan:~% getfacl verzeichnis3
# file: verzeichnis3
# owner: stefan
# group: users
user::rwx
user:stka:rwx
group::r-x
group:cdrom:rwx
mask::rwx
other::r-x
```

Wenn Sie jetzt in dem Verzeichnis eine neue Datei oder Verzeichnis erstellen, werden Sie feststellen, dass die ACLs nicht auf die neuen Einträge übernommen wurden. Damit die ACLs auf neue Einträge vererbt werden, müssen Sie die "defaults ACLs" an Verzeichnissen setzen.

Die default ACLs

Bis jetzt waren alle ACLs immer nur für den Eintrag gültig, auf dem Sie die ACL gesetzt hatten. Mit den default ACLs können Sie jetzt Berechtigungen auf ein Verzeichnis setzen, die sich dann an alle darunter neu erstellten Einträge vererben. Das Listing zeigt wie Sie default ACLs setzen können:

```
stefan@stefan:~% mkdir verz-def-acl

stefan@stefan:~% setfacl -d -m  g:cdrom:rwx verz-def-acl

stefan@stefan:~% ls -ld verz-def-acl
drwxr-xr-x+ 2 stefan users 4096 Aug  6 10:57 verz-def-acl

stefan@stefan:~% getfacl verz-def-acl
```

```
# file: verz-def-acl
# owner: stefan
# group: users
user::rwx
group::r-x
other::r-x
default:user::rwx
default:group::r-x
default:group:cdrom:rwx
default:mask::rwx
default:other::r-x
```

Mit der setfacl-Option "-d" erstellen Sie ein default ACL. Wenn Sie sich das Verzeichnis mit "ls" anzeigen lassen, sehen Sie auch hier wieder das Pluszeichen am Ende der Rechteliste, welches auf das Vorhandensein von ACL hinweist. Lassen Sie sich jetzt die ACLs mittels "getfacl" anzeigen, sehen Sie, dass alle ACLs als "default ACLs" eingetragen wurden. Wenn Sie jetzt in dem Verzeichnis ein neues Verzeichnis oder eine neue Datei anlegen, werden diese Einträge die ACLs übernehmen. In nächsten Listing möchte ich Sie noch einmal auf das Verhalten beim Anlegen einer Datei hinsichtlich des x-Rechts aufmerksam machen. In den "default ACLs" ist ja das x-Recht immer mit gesetzt und wird somit auf alle neuen Einträge übernommen. Auch auf Dateien werden diese "default ACLs" gesetzt. Wenn Sie jetzt aber eine leere Datei oder eine Datei einer Anwendung erstellen, vergibt das Betriebssystem das x-Recht nicht an diese Datei. In den ACLs ist es aber gesetzt, deshalb sehen Sie bei dem Aufruf von "getfacl" den Hinweis "effective". Das ist kein Fehler, das ist das richtige Verhalten.

```
stefan@stefan:~% touch dat-acl

stefan@stefan:~% getfacl dat-acl
# file: dat-acl
# owner: stefan
# group: users
user::rw-
group::r-x                      #effective:r--
group:cdrom:rwx                 #effective:rw-
mask::rw-
other::r--
```

Löschen von ACLs

Sowohl die einfachen, als auch die default ACLs löschen Sie mit dem Kommando "setfacl". Beim Entfernen der ACLs wird zwischen den unterschiedlichen ACLs unterschieden. Sie müssen vor dem Löschen wissen, ob Sie ein einfache oder eine default ACL entfernen wollen. Das folgende Listing zeigt den Vorgang des Löschens einer default ACL:

```
stefan@stefan:~/verz-def-acl% setfacl -k verz-acl

stefan@stefan:~/verz-def-acl% getfacl verz-acl
# file: verz-acl
# owner: stefan
```

```
# group: users
user::rwx
group::r-x
group:cdrom:rwx
mask::rwx
other::r-x

stefan@stefan:berechtigungen-ia/verz-def-acl% ls -ld verz-acl
drwxrwxr-x+ 2 stefan users 4096 Aug  6 11:05 verz-acl
```

Durch die Option "-k" werden alle default ACLs des Eintrags gelöscht, die einfachen ACLs bleiben erhalten. Mit der Option "-b" löschen Sie die einfachen ACLs:

```
stefan@stefan:~/verz-def-acl% setfacl -b verz-acl

stefan@stefan:~/verz-def-acl% ls -ld verz-acl
drwxr-xr-x 2 stefan users 4096 Aug  6 11:05 verz-acl
```

Und dann ist da noch das Folgende

Bis zu diesem Zeitpunkt haben wir immer nur ACLs auf einzelne Einträge gesetzt oder entfernt. Selbstverständlich können Sie die ACLs auch rekursiv über einen ganzen Teilbaum setzen oder löschen. Auch ein Sichern der ACLs mittels "getfacl" können Sie rekursiv durchführen und diese Sicherung später mittels "setfacl" wieder einspielen. Das ist besonders dann wichtig, wenn Sie eine Backup-Software einsetzen, die keine ACLs sichern kann. Dann können Sie die ACLs getrennt in eine Datei sichern und anschließend – bei einem eventuellen Recovery des Dateisystems – aus der Datei wieder einspielen. Sowohl das Kommando "setfacl" als auch das Kommando "getfacl" kennen die Option "-R" für das rekursive Löschen oder Anzeigen. Im folgenden Listing sehen Sie ein paar Beispiele für die Verwendung von "setfacl" und "getfacl":

```
stefan@stefan:~% getfacl -R verz-def-acl

stefan@stefan:~% getfacl -R verz-def-acl > sicherung.acl

stefan@stefan:~% setfacl -R -m g:cdrom:rwx verz-def-acl

stefan@stefan:~% setfacl --restore=sicherung.acl
```

Im ersten Beispiel werden alle ACLs aller Einträge angezeigt. Im zweiten Beispiel werden die Einträge in eine Datei gesichert. Beim dritten Beispiel werden alle Einträge ab dem angegeben Verzeichnis mit einer ACL belegt. Das vierte Beispiel zeigt dann, wie Sie die ACLs aus einer Datei wieder einspielen können. Wenn Sie jetzt die ACLs in Ihrem Dateisystem anwenden wollen, dann stellen Sie schnell fest, dass Dateien und Verzeichnisse unterschiedliche ACLs benötigen. Zum Beispiel braucht ein Textdokument kein x-Recht, aber ein Verzeichnis benötigt dieses Recht auf jeden Fall, denn sonst kann nicht in das Verzeichnis gewechselt werden. Deshalb können Sie das Kommando "setfacl" sehr gut zusammen mit dem Kommando "find" einsetzen. Im nächsten Listing zeige ich Ihnen, wie Sie für alle Dateien und Verzeichnisse ab einem bestimmten Punkt ACLs setzen

können:

```
stefan@stefan:~% find . -type d -exec setfacl -d -m g:cdrom:rwx {} \;

stefan@stefan:~% find . -type f -exec setfacl -m g:cdrom:r {} \;
```

Im ersten Beispiel werden alle Verzeichnisse ab dem aktuellen Verzeichnis gesucht und anschließend wird für jedes Verzeichnis die entsprechende default ACL gesetzt. Im zweiten Beispiel wird nach allen Dateien gesucht und die einfache ACL gesetzt. Die Option "-d" darf hier nicht verwendet werden, da sich default ACLs nur auf Verzeichnisse setzen lassen. Verwenden Sie trotzdem die Option "-d", werden für die Dateien Fehlermeldungen ausgeworfen.

Zusammen mit den Berechtigungen, den "special-Bits" und den ACLs stehen Ihnen jetzt eine Vielzahl von Möglichkeiten zur Verfügung, die Rechte in Ihrem Dateisystem gezielt zu setzen.

Die erweiterten Attribute

Auch Linux kennt im Dateisystem erweiterte Attribute mit denen Sie bestimmte Einschränkungen beim Zugriff auf Dateien einrichten können. Diese erweiterten Attribute möchte ich an dieser Stelle vorstellen. Um die Attribute verwenden zu können muss das Dateisystem vorbereitet werden. Denn auch für die Attribute gibt es eine Mountoption, die Sie wieder bei einigen älteren Distributionen in der Datei "/etc/fstab" setzen müssen. Das Listing zeigt Ihnen den angepassten Eintrag in der Datei:

```
/dev/hdb1       /daten              ext4    errors=remount-ro,acl,user_xattr 0       0
```

Nach einem "remount" des Dateisystems stehen Ihnen dann die erweiterten Attribute zur Verfügung. Die Attribute setzen Sie mit dem Kommando "chattr". Auflisten können Sie die Attribute mit dem Kommando "lsattr", sollten diese Kommandos auf Ihrem System nicht vorhanden sein, müssen Sie das Paket "attr" nachinstallieren.

Bei den Attributen gibt es zwei Gruppen: Die Attribute der einen Gruppe können von "normalen" Benutzern gesetzt werden, die Attribute der anderen Gruppe können nur vom "root" gesetzt werden. Unter den Attributen gibt es einige, die sowieso selten benötigt werden, diese sollen hier nicht betrachtet werden. Aber einige der Attribute können im täglichen Umgang mit Dateisystemrechten nützlich sein. Diese werden ich Ihnen hier erklären.

Attribute, die jeder setzen kann

- **Das Attribut "d"**
  Durch Setzen dieses Attributs verhindern Sie, dass die entsprechende Datei bei einer Datensicherung mit dem Kommando "dump" mitgesichert wird.
- **Das Attribut "s"**
  Wenn ein Benutzer dieses Attribut setzt, wird die Datei beim Löschen mit Nullen überschrieben.

- **Das Attribut "A"**
  Wenn Sie das Attribut "A" setzen, wird die "atime" der Datei nicht verändert. Das kann die Performance beim Schreiben erhöhen. Besser ist es aber, hier die Option "noatime" für das gesamte Dateisystem in der Datei "/etc/fstab" zu setzen.

Attribute die nur der "root" setzen kann

- **Das Attribut "a"**
  Wenn Sie als "root" dieses Attribut auf eine Datei setzen, kann der Inhalt der Datei nicht mehr geändertwerden, es können nur noch Daten an die Datei angehängt werden. Hier sehen Sie dazu ein Beispiel:

  ```
  root@stefan:~#  touch datei1.txt root@stefan:~#  chattr +a datei1.txt  root@stefan:~#  ls
  ```

  Mit dem ersten Kommando wird einfach eine leere Datei erzeugt, bei der im zweiten Schritt das Attribut "a" gesetzt wird. Im Anschluss wird versucht, eine Zeile in die Datei mit einer einfachen Umleitung zu schreiben. Diese Operation wird aufgrund des Attributs verboten. Selbst der "root" kann den Inhalt der Datei nicht verändern, solange das Attribute "a" gesetzt ist. Erst im nächsten Schritt wird eine Zeile an die Datei angehängt, und jetzt ist das
  Schreiben in die Datei möglich.

- **Das Attribut "i"**
  Durch das Attribut "i" wird die Datei immun gegen das Ändern, Umbenennen und Löschen. Auch der "root" kann eine Datei mit diesem Attribut nicht löschen, ohne vorher das Attribut zurückzusetzen. Auch dazu sehen Sie ein Beispiel: root@stefan:~# chattr +i datei1.txt

  ```
  root@stefan:~# lsattr datei1.txt  ----i-------------- datei1.txt root@stefan:~# echo "Ein
  ```

Alle Aktionen wurden hier als Benutzer "root" durchgeführt. Wie Sie sehen, ist danach keine Aktion mit der Datei mehr möglich. Erst wenn das Attribut von der Datei entfernt wird, kann die Datei wieder gelöscht werden.

Weitere Attribute

Neben diesen Attributen gibt es noch weitere Attribute, die zum Teil experimentell oder nicht von so großer Bedeutung für den täglichen Gebrauch sind. Alle Attribute werden ausführlich in der Manpage zu "chattr" beschrieben. Wenn Sie die Attribute verwenden wollen, sollten Sie immer einen Blick auf diese Manpage werfen, um mögliche Komplikationen zu vermeiden, denn nicht alle Attribute funktionieren in allen Dateisystemen.

Ein praktisches Beispiel

Nach dem im ersten Teil die Grundlagen besprochen wurden, will ich Ihnen jetzt anhand eines Beispiels zeigen, wie Sie die Dateisystemrechte planen können. Natürlich ist es in so einem Artikel nicht möglich, alle Eventualitäten abzudecken, aber eine gewisse Planungsgrundlage kann ich Ihnen hier an die Hand geben. In dem Beispiel werde ich eine Verzeichnisstruktur erstellen, die sich auf ein Unternehmen mit mehreren Abteilungen bezieht, in dem jede Abteilung unterschiedliche Zugriffsrechte auf verschiedene Ordner benötigt. Desweiteren soll ein Verzeichnis für alle Mitarbeiter existieren, das als Austauschverzeichnis für Daten dient und in das alle Mitarbeiter schreiben dürfen, aber nur der Besitzer einer Datei soll diese auch löschen können. Um die Verzeichnisse hier abbilden zu können, werde ich das Kommando "tree" verwenden. Sollte das Kommando bei Ihnen nicht installiert sein, müssen Sie das Paket "tree" nachinstallieren.

Im Beispiel wird alles auf einer lokalen Maschine eingerichtet. In der Praxis werden Sie die Daten auf einem Server verwalten und die Benutzerverwaltung zentral durchführen. Da sich die ACLs und die Dateisystemrechte via NFS auf Clients exportieren lassen, können Sie das Beispiel auch auf einem Fileserver mit einer zentralen Benutzerverwaltung und angebundenen Clients einrichten und testen.

Beschreibung der Umgebung

Die Firma besteht aus drei Abteilungen: Der Verwaltung, der Produktion und der Geschäftsleitung. Jede Abteilung soll ein eigenes Verzeichnis erhalten, in der nur diese Abteilung Schreibrechte haben soll. Die Geschäftsleitung möchte auf allen Abteilungsverzeichnissen das Leserecht haben. Alle Dateien in den Abteilungsverzeichnissen sollen immer der Abteilungsgruppe gehören. Die Rechte aller neuen Einträge sollen immer mit der Umask 007 angelegt werden. Es soll ein Gruppe "mitarbeiter" geben, in der alle Mitarbeiter Mitglied sind. Über die Gruppe sollen die Rechte am Austauschverzeichnis gesteuert werden.

Setzen der Umask

Damit alle Mitarbeiter auf allen Clients die gewünschte Umask von 007 erhalten, müssen Sie, je nach Distribution, entweder direkt die Datei "/etc/profile" oder die Datei "/etc/login.defs" anpassen. Diese Anpassung müssen Sie an jedem Client durchführen.

Einrichten der Verzeichnisstruktur

Um die Berechtigungen gut staffeln zu können, wird ein übergeordnetes Verzeichnis für die Abteilungen eingerichtet. An diesem Verzeichnis erhalten alle Mitarbeiter über die Gruppe "mitarbeiter" die Rechte "rx", damit sie durch das Verzeichnis in das eigentliche Abteilungsverzeichnis wechseln können. Das wäre auf einem Server auch das Verzeichnis, das über NFS an alle Clients freigegeben würde. Das folgende Listing zeigt das Anlegen der Verzeichnisstruktur:

```
root@stefan:~# mkdir  /abteilungen
```

```
root@stefan:~# cd /abteilungen

root@stefan:/abteilungen# mkdir verwaltung produktion geschaeftsleitung

root@stefan:/abteilungen# mkdir /alle
```

Nach dem alle benötigten Verzeichnisse angelegt wurden, zeigt das nächste Listing, wie Sie die entsprechenden Rechte setzen müssen:

```
root@stefan:/# chmod 750 /abteilungen

root@stefan:/# chmod 2770 /abteilungen/*

root@stefan:/# chgrp mitarbeiter /abteilungen

root@stefan:/# chgrp verwaltung /abteilungen/verwaltung

root@stefan:/# chgrp produktion /abteilungen/produktion

root@stefan:/# chgrp geschaeftsleitung /abteilungen/geschaeftsleitung

root@stefan:/# tree -p abteilungen
abteilungen
--- [drwxrws---]  geschaeftsleitung
--- [drwxrws---]  produktion
--- [drwxrws---]  verwaltung

root@stefan:/# chgrp mitarbeiter /alle

root@stefan:/# chmod 3770 /alle

root@stefan:/# ls -ld /alle
drwxrws--T 2 root mitarbeiter 4096 Aug 10 13:04 /alle
```

Jetzt kann jede Abteilung in ihrem Verzeichnis Daten speichern. Durch das gesetzte SGID-Bit gehören alle Dateien immer der Abteilungsgruppe. Über "other" werden keine Rechte vergeben. Am Verzeichnis "/alle" haben alle Mitarbeiter die Rechte "rwx". Dadurch können sie Einträge erzeugen und Inhalte ändern. Aber nur die Einträge, die ihnen gehören können sie löschen oder umbenennen. Jetzt fehlt nur noch, dass die Geschäftsleitung an allen anderen Abteilungsverzeichnissen das Leserecht an allen Einträge erhält. Das steuern Sie über die default ACLs an den Verzeichnissen. Das Listing zeigt die entsprechenden Kommandos:

```
root@stefan:/# setfacl -d -m g:geschaeftsleitung:rx /abteilungen/verwaltung

root@stefan:/# setfacl -d -m g:geschaeftsleitung:rx /abteilungen/produktion
```

Jetzt können alle Mitglieder der Gruppe "geschaeftsleitung" in allen Verzeichnissen mindestens lesen. Dadurch, dass die ACLs als default ACLs gesetzt wurden, vererben sich die ACLs immer weiter und auch in neu angelegten Unterverzeichnissen hat die Geschäftsleitung die gewünschten

Rechte.

Fazit

Ich hoffe, ich konnte Ihnen mit diesem kleinen Artikel zum Thema Dateisystemberechtigungen etwas helfen, in Zukunft einfacher Verzeichnisstrukturen für Mitarbeiter bereitzustellen und diese mit den benötigten Rechten und ACLs zu versehen.

Autor

Publikationen

- Shell-Programmierung: Das umfassende Handbuch
- Linux-Server: Das Administrationshandbuch
- Linux-Server: Das umfassende Handbuch
- Samba 4: Das Handbuch für Administratoren

Revision #2
Created 2026-03-18 22:21:12 UTC by willi
Updated 2026-03-18 22:24:06 UTC by willi