Jörg Reinholz, fastix WebDesign & Consult, Kassel

Benutzerrechte:

Filesystem Access-Control-List (ACL) unter Linux

Inhaltsverzeichnis

Über den Autor:

Jörg Reinholz ist seit dem Jahr 1999 als bundesweit IT-Trainer und Privatdozent tätig. Seit dem Jahr 2002 bearbeitet er Linux- und Unix-Themen (Grundlagen. Programmierung, Server, LPIC I und II) , Themen im Bereich WebDesign (von HTML über CSS, Javascript, PHP bis zum Apache Webserver), Datenbanken (MySQL, Maria DB) und Programmierung (und, ach, auch Perl!)

Nur ausnahmsweise befasst sich Jörg Reinholz mit schönen „Windows“ Themen wie Excel-VBA.

Teilnehmer seiner Seminare sind Mitarbeiter von kleinen und großen (Dax-)Unternehmen oder Organisationen, wie z.B. die Bundeswehr, das österr. Heer oder die Fachhochschule Vorarlberg.

Jörg Reinholz berät auch Unternehmen, programmiert, pflegt Server und wirkt an Webauftritten mit..

Vorwort:

Immer wieder höre ich, dass Linux bei der Vergabe von Benutzerrechten Windows nachstehe. Der Grund hierfür ist jedoch nicht, dass Linux Benutzerrechte an Dateien nicht genau so genau vergeben kann wie Windows, der Grund ist, dass zu viele nicht wissen wie das geht. Und zugegeben: es gibt derzeit für einiges hiervon keine grafische Oberfläche.

Der Mangel an , nennen wir es „anklickbaren“ war es dann aber auch schon, für Administratoren dürfte schon wesentlich interessanter sein, dass sich Vorlagen für Benutzerrechte in menschen- und maschinenlesbaren Textdateien speichern und also auch verarbeiten lassen. Diese Dimension bietet Windows nicht.

Da ich damit rechnen muss das diesen Artikel auch (künftige) Systemadministratoren lesen werden die bisher nur wenig mit Linux zu tun hatten beginne ich mit einem Exkurs in die „gewöhnlichen“ Rechteverwaltung unter Unix.

Wer das nicht zu brauchen glaubt, der lese gleich auf Seite 8 weiter...

Exkurs: Die Unix-Standard-Benutzerrechte (Grundlagen)

Für die Berechtigung an Dateien und Verzeichnissen gibt es folgende Befehle:

Die Befehle chown, chgrp

Beide legen den Eigentümer („owner“, benannt als „user“- das wird
später wichtig) der Dateien fest. Unter Windows entspricht der „owner“ alias „user“ dem Autor.

Dabei gelten folgende Regeln:

1. Einfache Benutzer dürfen Dateien oder Verzeichnisse keinem anderen Benutzer verschenken. Dies darf der Administrator (root).
2. Einfache Benutzer dürfen Dateien oder Verzeichnisse nur einer Gruppe zuordnen, der sie selbst angehören.
3. Einfache Benutzer dürfen nur die Rechte an Dateien oder Verzeichnissen nur einer Gruppe zuordnen, der sie selbst angehören.

Syntaxbeispiele:

```
~> chown benutzer:gruppe datei
```

... Die Datei „datei“ wird dem Eigentümer „benutzer“ und der Gruppe „gruppe“ zugeordnet.

```
~>  chown benutzer datei
```

... Die Datei „datei“ wird dem Eigentümer „benutzer“ zugeordnet. Die Gruppe bleibt unverändert.

```
~>  chown :gruppe datei
```

... Die Datei „datei“ wird der Gruppe „gruppe“ zugeordnet. Der Eigentümer bleibt unverändert.

```
~>  chgrp gruppe datei
```

... Die Datei „datei“ wird der Gruppe „gruppe“ zugeordnet. Der Eigentümer bleibt unverändert. Ja, das ist im Prinzip das gleiche, man muss sich einen Befehl mehr merken und spart dafür einen Doppelpunkt...

Der Befehl chmod

Mit diesem Befehl lassen sich die Rechte für Ersteller bzw. Eigentümer („author“), die Gruppe („group“) und die ganze Welt („others“) vergeben.

Es gibt dafür zwei Schreibweisen, zuerst sollte jedoch anhand der folgenden Tabelle ein Überblick genommen werden, welche Rechte es gibt:

angewendet auf ► Recht ▼	**Datei**	**Verzeichnis**	**„Wert“**	**Symbol**
Lesen	darstellen, lesen	Inhalt auflisten	4	r
Schreiben	Verändern, Leeren	Dateien und Verzeichnisse im Verzeichnis anlegen, löschen, umbenennen	2	w
Ausführen	Ausführen	Betreten	1	x

(Tabelle: Die grundlegenden Unix/Linux-Rechte)

Für den Befehl „chmod“ gibt es zwei Schreibweisen

Das Zahlen/Punkte-System anhand eines Beispiels:

- Es soll das Recht die Datei auszuführen an den Benutzer/Eigentümer, Gruppe und die ganze Welt vergeben werden.
- Nur der Eigentümer soll die Datei ändern können.

Hierzu werden einfach die „Werte“ für die Benutzergruppen addiert.

Der erste zu vergebende Wert ist für den „user“ (Eigentümer, Ersteller):

Um die Datei ausführen zu können muss er sie auch lesen können, also:
Lesen (4) + Verändern (2) + Ausführen (1) . Das sind dann 7 Punkte.

Die Gruppe und Jedermann:

Lesen (4) + Ausführen (1). Das sind dann je 5 Punkte.
Die zu bildende Folge ist „user“ (7), „group“ (5), „others“ (5). Ergo „755“.

Der Befehl lautet also:

```
benutzer@host~>chmod 755 datei
```

Das ist einfach.

Das Symbol-System:

Für den Eigentümer/Autor steht das „u“- wie „user“. Für die Gruppe ein „g“ - wie „group“. Für die Welt ein „o“ wie „others“. Das wird gleich gebraucht. Die Rechte lassen sich dann mit „+“ und dem Symbol hinzufügen oder mit dem „-“ entziehen:

```
benutzer@host~> chmod u+rwx datei
```

gibt dem Eigentümer alle Rechte (Lesen, Schreiben, Ausführen) an der Datei „datei“.

```
chmod go+rx datei
```

gibt der Gruppe und der Welt die Rechte Lesen und Ausführen an der Datei „datei“.

```
benutzer@host~> chmod go-x datei
```

entzieht der der Gruppe und der Welt das Recht diese Ausführen. Falls Sie es je hatten.

Auch das war einfach.

Der Befehl umask

- angewendet auf die aktuelle Benutzersitzung oder Shell legt umask fest, welche Rechte künftig(!) in der gleichen Sitzung oder Shell (oder deren Kindern) neu angelegte Dateien oder Verzeichnisse vergeben werden. Hier kommt das Punkte-System zum Tragen. Es werden die Rechte, die nicht vergeben werden sollen, einfach abgezogen.

Die Verwendung von umask anhand eines Beispiels:

Es sollen künftig die Rechte an neuen Dateien und Verzeichnissen wie folgt vergeben werden:

- Der Eigentümer soll Dateien lesen, schreiben, Verzeichnisse betreten, listen, darin Dateien anlegen und löschen dürfen;
- die Gruppe: Dateien Lesen, Verzeichnisse betreten und listen;
- der Rest der Welt soll keine Rechte an Dateien und Verzeichnissen erhalten.

Vorgehen:

- Der Eigentümer bekommt einen Wert von 0: Es wird ihm kein Recht entzogen (Das Ausführen-Recht wird für Dateien nicht automatisch vergeben. Dies ist kein Windows.
- Die Gruppe bekommt einen Wert von 2: Es wird das Schreibrecht entzogen (Das Ausführen-Recht wird für Dateien nicht automatisch vergeben. Dies ist kein Windows.
- Die Welt bekommt 7: Es wird das Lesen, Schreiben und Ausführen/Betreten verboten.

Der Befehl lautet also

benutzer@host~> umask 027

1. Test durch anlegen und prüfen einer Datei:

```
benutzer@host~> touch foo && ls -l foo
```

sollte folgendes ausgeben:

```
-rwxr-x--- benutzer users 0 25. Jun 23:53 foo
```

2. Test durch anlegen und prüfen eines Verzeichnisses:

```
benutzer@host~> mkdir bar && ls -ld bar
```

sollte folgendes ausgeben:

```
drwxr-w--- 2 benutzer users  4096 25. Jun 23:55 bar
```

War einfach- oder?

Tip!

Es ist eine gute Idee unter Linux in der Datei ~/.profile Folgendes einzutragen:

```
umask 077
```

Das geht übrigens am schnellsten mit:

```
echo 'umask 077' >> ~/.profile
```

Das hat geklappt, wir sind wir fit für die Access-Control-List!

Acess-Control-Lists (ACL) unter Linux

Vorbereitung der Beispiele / Einstellungen des Systems

Für das folgende Vorgehen brauchen wir, ein paar ganz „normale“ Benutzer und Gruppen:

Der Benutzer „user“ (uid:1000) ist Mitglied in den Gruppe

- users (gid: 100)

Der Benutzer „anna“ (uid:1021) ist Mitglied in den Gruppen

- users (gid: 100)
- verwaltung (gid: 1003)
- azubis (gid: 1004)

Der Benutzer „maria“(uid:1022) ist Mitglied in den Gruppen

- users (gid: 100)
- verwaltung (gid: 1003)

Der Benutzer „jonas“ (uid:1023) ist Mitglied in den Gruppen

- users (gid: 100)
- produktion (gid: 1002)
- azubis (gid: 1004)

Der Benutzer „oswald“ (uid:1024) ist Mitglied in den Gruppen

- users (gid: 100)
- produktion (gid: 1002)

Wie diese Benutzer anzulegen sind erklärt „man useradd“ und „man usermod“.

Außerdem sollte sichergestellt sein, dass das Paket „acl“ installiert ist. Dieses gehört aber bei modernen Distributionen zum Standard. Wenn nicht hilft, je nach Distribution einer der folgenden Befehle:

```
sudo apt-get install acl
sudo yum -install acl
sudo zypper -i acl
```

Es geht hier aber nicht um das Paketmanagement oder Softwareinstallation.

Wir legen als Benutzer „user“ im Ordnet /tmp einen Unterordner „test“ an, vergeben für jedermann das Recht den Ordner zu betreten und zu lesen und für den „user“ zu schreiben und wechseln gleich in diesen Ordner:

```
user@maschine: ~> mkdir /tmp/test
user@maschine: ~> chmod 755 /tmp/test
user@maschine: ~> cd /tmp/test
```

Um nicht zu viele Rechte zu vergeben legen wir jetzt fest, dass ab sofort nur der Eigentümer Rechte bekommt:

```
user@maschine: ~> umask 077
```

Jetzt legen wir eine Datei an:

```
user@maschine ~> touch datei
```

Die „normalen“ Unix-Rechte an der Datei sehen wir mit „ll“ oder „ls -l“:

```
user@maschine:/tmp/test> ls -l datei

-rwx------ 1 user users 0 26. Jun 12:36 datei
```

Die Rechte sind also wie vorgesehen vergeben.

ACL einsehen

Die ACL sehen wir mit „ getfacl [verzeichnis/]dateiname “:

```
user@maschine:/tmp/test> getfacl datei
# file: datei
# owner: user
# group: users
user::rw
group::
other::
```

Mit „user“ ist hier übrigens noch immer der „Autor“ oder „Eigentümer“ gemeint. Nicht der Benutzer „user“. Das kann verwirren.

ACL setzen

Das ist doch sehr übersichtlich? Oder? Wenn „getfacl“ die „Access Control List“ ausgibt, dann liegt nahe, dass „setfac“ diese setzt. Wenn nur die Syntax nicht wäre. Aber auch die ist nicht wirklich schwer:

Ein: setfacl -m ACL datei ODER setfacl --modify=“ACL“ datei
setzt die Rechte wie in ACL festgelegt.

Beispiel: anna soll Leserechte erhalten:

```
user@maschine:/tmp/test>setfacl -m anna:r datei
user@maschine:/tmp/test> getfacl datei

# file: datei
```

```
# owner: user
# group: users
user::rwx
user:anna:r--
group::---
other::---
```

Aha! Anna hat also Lese-Rechte bekommen. Mit:

```
user@maschine:/tmp/test> setfacl --modify anna:rw datei
```

hätte sie auch Schreibrechte. Das ließe sich beliebig wiederholen.
Jetzt wollen wir die Rechte in einer Datei („muster.acl“) festhalten:

user@maschine:/tmp/test> getfacl datei > muster.acl

Das ist eine tolle Sache, denn die Rechte lassen sich auch aus dieser Datei lesen und vergeben:

```
user@maschine:/tmp/test> touch datei2
user@maschine:/tmp/test> setfacl -M muster.acl datei2
```

Ein:

```
user@maschine:/tmp/test> cat muster.acl
```

liefert:

```
# file: datei
# owner: user
# group: users
user::rw-
user:anna:r--
group::---
other::---
```

Verwenden wir das:

```
user@maschine:/tmp/test> setfacl -M muster.acl  datei2
```

ODER

```
user@maschine:/tmp/test> setfacl --modify --file=muster.acl  datei2
```

setzt die Rechte, die wir uns jetzt ansehen:

```
user@maschine:/tmp/test> getfacl datei2

# file: datei2
# owner: user
# group: users
user::rw-
```

```
user:anna:r--
group::---
other::---
```

Wie merken uns: Rechte werden mit „setfacl -m“ oder „setfacl -M“ gesetzt. Bei -M müssen wir eine Textdatei angeben, die die Rechte enthält, wie sie mit „getfacl“ ausgegeben werden.

Das geht auch direkt:

```
user@maschine:/tmp/test> touch datei3
user@maschine:/tmp/test> getfacl datei | setfacl --set-file=- datei3
user@maschine:/tmp/test> getfacl datei3
# file: datei3
# owner: user
# group: users
user::rw-
user:anna:r--
group::---
other::---
```

Hier sorgt das „-“ nach „--set-file“, das die Daten von <stdin> gelesen werden. Da schreibt „getfacl“ die Daten über das Pipe herein.

Das war nicht wirklich schwer. Man kann ja einen kleinen Batch schreiben:

Tipp:

Mit:

```
touch ~/bin/cpACL
chmod 755 ~/bin/cpACL
# Hier wird noch chmod verwendet :)
vi ~/bin/cpACL
```

Inhalt:

```
#! /bin/sh
# Setzt die ACL von $1 nach $2.
# Hier können noch Prüfungen eingebaut werden:
echo Kopiere Rechte von Datei $1 nach Datei $2
getfacl $1 | setfacl --set-file=- $2
echo '#################### Ergebnis: ########################'
getfacl $2
echo '######################################################'
```

Doch wie ist das mit Gruppen?

Einfach! Wie sonst?

```
user@maschine:/tmp/test> setfacl -m g:produktion:r datei
user@maschine:/tmp/test> getfacl datei
```

Ergebnis:

```
# file: datei
# owner: user
# group: users
user::rw-
user:anna:r--
group::---
group:produktion:r--
other::---
```

Die Gruppe „produktion“ hat das Leserecht erhalten.

Jetzt schauen wir mal weiter:

Anna erhält Lese- und Schreibrechte, die Gruppe „verwaltung“ Leserechte:

```
user@maschine:/tmp/test> touch datei4
user@maschine:/tmp/test> setfacl -m g:verwaltung:r -m anna:rw datei4
```

Ein „getfacl datei4“ wird uns zeigen, dass dies funktioniert hat.

Rechte entziehen

Mit „setfacl -b datei“ oder „setfacl --remove datei“ all löschen wir alle ACLs. Die gewöhnlichen Nutzerrechte bleiben aber.

```
user@maschine:/tmp/test> setfacl -b datei
user@maschine:/tmp/test> getfacl datei

# file: datei
# owner: user
# group: users
user::rw-
group::---
other::---
```

Jetzt sind die Rechte wieder weg. Einfach weg.

An „datei3“ hatte Anna Leserechte. Wir können Annas Sonderrechte komplett entziehen:

```
user@maschine:/tmp/test> setfacl x anna datei3
user@maschine:/tmp/test> getfacl datei

# file: datei3
# owner: user
# group: users
user::rw-
group::---
other::---
```

Auch an der Datei datei2 hatte Anna Leserechte. Sie erhält jetzt auch das Schreibrecht. Das können wir dannauch einzeln entziehen, indem wir die Rechte neu setzen:

```
user@maschine:/tmp/test> setfacl -m anna:rw datei3
user@maschine:/tmp/test> getfacl datei

# file: datei3
# owner: user
# group: users
user::rw-
user:anna:rw-
group::---
other::---
```

```
user@maschine:/tmp/test> setfacl -m anna:r datei3
user@maschine:/tmp/test> getfacl datei
# file: datei3
# owner: user
# group: users
user::rw-
user:anna:r--
group::---
other::---
```

Ich glaube, auch das war nicht schwer.

Jetzt sollte uns niemand hindern, an allen Dateien (und Verzeichnissen) im aktuellen Ordner alle ACLs zu löschen:

```
user@maschine:/tmp/test> setfacl -b *
```

Jetzt sind sie weg... wie ein „getfacl“ zeigen würde. Wir können also auch Dateilisten übergeben.

Dateien in Unterverzeichnissen

Auch das geht komfortabel. Legen wir ein paar an:

```
user@maschine:/tmp/test> mkdir verz1
user@maschine:/tmp/test> mkdir verz2
user@maschine:/tmp/test> touch verz1/datei
user@maschine:/tmp/test> touch verz2/datei
```

Setzen wir die Rechte:

```
user@maschine:/tmp/test> setfacl -R -m anna:rx verz1
user@maschine:/tmp/test> setfacl -R -m anna:rwx verz2
```

Lesen wir die Rechte:

```
user@maschine:/tmp/test> getfacl verz1

# file: verz1
# owner: user
# group: users
user::rwx
user:anna:r-x
group::---
mask::r-x
other::---
```

```
user@maschine:/tmp/test> /tmp/test> getfacl -R verz1

# file: verz1
# owner: user
# group: users
user::rwx
user:anna:r-x
group::---
mask::r-x
other::---

# file: verz1/datei
# owner: user
# group: users
user::rw-
user:anna:r-x
group::---
mask::r-x
other::---
```

Vorsicht! Genau wie chmod setzt setfacl auch Ausführen-Rechte für Dateien!

mit

```
user@maschine:/tmp/test> setfacl -R -m anna:rw  verz2/*
```

sollten wir diese wieder entziehen!

ACL sichern und wiederherstellen

mit

```
user@maschine:/tmp/test> getfacl -R * > dateirechte.backup
```

können die ACL aller Dateien und Verzeichnisse und Unterverzeichnisse und der Dateien in diesen gesichert werden. Mit

```
user@maschine:/tmp/test> setfacl --restore=dateirechte.backup
```

stellt man diese wieder her.

Dabei ist folgendes zu beachten:

- Enthält die Datei als „Kommentar“ Benutzer- und Gruppeninformationen zu den Eigentümern der Dateien und Verzeichnisse und wird der Befehl von „root“ ausgeführt, dann werden auch diese Eigenschaften wieder hergestellt.
- Führt ein normaler Benutzer den Befehl aus, dann wird dieser natürlich nur im Rahmen seiner (aktuellen) Rechte ausgeführt.

ACL als Voreinstellungen

Auch dies ist möglich:

```
user@maschine:/tmp/test> setfacl -m d:anna:rw -m d:g:produktion:r
```

Es wird also keine Datei und auch kein Verzeichnis addressiert! Dennoch werden diese Regeln beachtet. In der Zukunft nämlich. Jedenfalls so lange Sie in der selben Sitzung arbeiten.

```
user@maschine:/tmp/test> touch datei7
user@maschine:/tmp/test> getfacl datei7
# file: datei7
# owner: user
# group: users
user::rw-
user:anna:rw-
group::r-x                      #effective:r--
group:produktion:r--
mask::rw-
other::r--
```

ACL leicht erkennen

Das Vorhandensein von ACLs erkennt mit „ls l“ oder „ll“:

```
user@maschine:/tmp/test> ls -l datei7
-rw-rw-r--+ 1 fastix users 0 26. Jun 14:50 datei7
```

Hinter den normal angegebenen Rechten steht ein „+“ wie für Wissende leicht zu erkennen ist. Genau das „+“ zeigt: Für diese Datei gelten ACL!

ACL komplex vergeben

Hierfür ist am besten eine Textdatei geeignet, in welcher die Rechte festgehalten sind.
Für unser obiges Beispiel soll folgendes Szenario gelten:

- Die (Standard-) Gruppe „users“ soll im aktuellen Verzeichnis und darin enthaltenen Dateien (auch
- in Unterordnern) nichts dürfen.
- Die Gruppe „produktion“ soll im aktuellen Verzeichnis lesen und schreiben dürfen.
- Die Gruppe „verwaltung“ soll im aktuellen Verzeichnis lesen dürfen.
- Die Gruppe „azubis“ soll im aktuellen Verzeichnis nichts dürfen.
- Die Benutzerin „anna“ soll im aktuellen Verzeichnis zusätzlich schreiben dürfen

Diese sollen auch die Voreinstellungen für das Verzeichnis werden. Wir schreiben also in die Datei „acl.liste“:

```
g:users:---
g:produktion :rw-
g:verwaltung :r--
g:azubis     :---
u:anna       :rw-
```

Für die Voreinstellungen müssen wir bedenken, dass diese auch für Unterverzeichnisse gelten sollen. Diese müssen betretbar werden. Wir schreiben also weiter in die Datei „acl.liste“:

```
d:g:users      :---
d:g:produktion :rwx
d:g:verwaltung :r-x
d:g:azubis     :---
d:u:anna       :rwx
```

Mit

```
setfacl -R --test -M acl.liste .
```

können wir das testen. (Achten Sie auf den Punkt für das aktuelle Verzeichnis!) Die Ausgabe ist ggf. sehr lang.
Ein

```
setfacl -R -M acl.liste .
```

macht die Einstellungen zur Wirklichkeit.

Hierbei ist es übrigens egal, ob laut „ls -l“ vorher Standard-Rechte vergeben wurden. Die ACL überschreibt diese.

ACL testen

Der Befehl „setfacl“ lässt sich unter allen Umständen mit der Option „--test“ ausführen und zeigt das mögliche Ergebnis an – ohne jedoch die Rechte wirklich zu setzen. Machen Sie davon Gebrauch und testen Sie Ihre Befehle oder Einstellungen!

Weiterführende Informationen

Bekommt man mit „man 1 setfacl“ und man „man 1 getfacl“.

Impressum

Autor und Inhaltlich verantwortlich für den Inhalt dieser Unterlage:

Jörg Reinholz, fastix WebDesign & Consult

Hafenstr. 67

34125 Kassel

Telefon: +49 561 3172277
Fax: +49 561 3172276
E-Mail: joerg.reinholz@fastix.org
Web: http://www.fastix.org

Lizenz: