Das Rechtesystem von Linux

Verständliche Einführung was Rechte sind, was sie bewirken und wie man diese verändert.

In der letzten Zeit häufen sich die Fragen in Foren welche das Rechtesystem betreffen. Leider verstehen Umsteiger hierbei oft nur Bahnhof und können sich die Fehlfunktionen welche daraus evtl. resultieren nicht erklären.

Deshalb diese Anleitung, die das Rechtesystem verständlich erklärt und einem Umsteiger diese „komplexe Materie“ näher bringen soll.

Gleich zum Anfang, es ist nicht so Komplex und Kompliziert wie es den Anschein hat. Man hat schnell begriffen wie man die Rechte in den Griff bekommt und diese nutzt.

Jeder Benutzer hat eine s.g. Benutzer- und Gruppennummer.

Inhalt:

1. Allgemeines

Sie werden sich nun fragen wozu muss ich mich damit beschäftigen bzw. wo kann ich auf meinem Linuxsystem mit den Rechten konfrontiert werden.

Nun, dies geht schon los wenn Sie eine externe Festplatte oder einen USB Stick einstecken und merken plötzlich das Sie zwar lesen, aber nicht auf das externe Gerät schreiben können.

Oder Sie haben sich ein Programm aus dem Internet heruntergeladen und möchten es nun installieren, aber das Programm startet nicht.

Also schauen wir uns an woran das liegen könnte.

Auf einem PC befinden sich Verzeichnisse und einzelne Dateien. Dementsprechend gibt es also

1. Verzeichnisrechte
2. Dateirechte

Ein Rechtesystem regelt nun, wer diese Dateien und Verzeichnisse

1. lesen
2. schreiben
3. ausführen

kann und darf (oder er hat keinerlei Rechte an dem jeweiligen Eintrag).

Dem Besitzer einer Datei obliegt es nun , wie er diese Rechte handhabt.

Aber Vorsicht, er kann sich natürlich auch selbst durch Veränderung dieser Rechte von einer Benutzung ausschließen, obwohl er der Eigentümer ist.

Wenn so etwas einmal passiert, keine Panik, der Superuser root (Administrator) kann in so einem Fall die Rechte wieder gerade biegen. (ja ich weis, es gibt auch Fälle in denen selbst Root dies nicht mehr kann, aber diese lassen wir hier einmal außen vor, in solch einem Fall kann man jedoch in der Regel dies mittels einer Live-CD auch in den Griff bekommen).

Noch kurz zu dem Begriff „Vererbung“. Damit ist gemeint, dass Rechte von einem Verzeichnis aus auf dessen untergeordnete Dateien und Verzeichnisse vererbt (also 1:1 weitergegeben) werden. Es werden also die gleichen Rechte, welches das übergeordnete Verzeichnis hat an dessen Inhalt (Dateien und Verzeichnisse) weitergegeben.

Ebenso kann ein Programm, auf andere Programme das es aufruft, seine Rechte weitergeben / vererben.

Wozu das nun alles, Linux (Unix) ist ein mehrplatzfähiges Betriebssystem (Multiuser System). Auch wenn es nur als Einzelplatzsystem genutzt wird, müssen die Rechte an Dateien und Verzeichnissen reglementiert (eingeschränkt) werden um ein gewisses Maß an Systemsicherheit zu bieten.

2. Datei- und Verzeichnisrechte

Wenn man in einem Terminal (Konsole) in seinem Homeverzeichnis z.b. einmal ein

```
ls -l    (kleines L)
```

ausführt, kann der Inhalt wie folgt aussehen:

```
drwxr-xr-x 5 cc    cc     4096 2010-09-17 07:33 Bilder
drwxr-xr-x 4 root root  4096 2010-10-23 14:41 Calibre Bibliothek
drwxr-xr-x 3 cc   cc     4096 2010-11-03 16:15 Calibre Library
drwxr-xr-x 4 cc   cc     4096 2010-11-10 18:50 Desktop
drwxr-xr-x 2 cc   cc     4096 2010-04-23 19:47 Dokumente
drwxr-xr-x 2 cc   cc     4096 2010-11-11 08:01 Downloads
```

Am Anfang jeder Zeile sehen wir eine Ansammlung von Buchstaben und Strichen gefolgt wieder von Buchstaben und Zahlen, bevor letztendlich der Name einer Datei oder des Verzeichnisses erscheint.

Diese Informationen geben uns den genauen Status einer Datei oder eines Verzeichnisses und sagen uns was und wer etwas mit einer Datei oder mit einem Verzeichnis machen kann und darf und was nicht.

Wir nehmen nun einmal eine Zeile Stück für Stück auseinander und schauen uns an was die Einträge für eine Bedeutung haben.

Zunächst einmal die ersten 10 Buchstaben und Strichkombinationen (gelesen wird von links nach rechts).

```
drwxr-xr-x 4 root root  4096 2010-10-23 14:41 Calibre Bibliothek
```

Der erste Eintrag steht entweder für:

d = directory (Verzeichnis) es handelt sich also um einen Verzeichniseintrag

- = würde bedeuten, dass es sich um eine normale Datei handelt.

c = character device (zeichenorientierte Gerätedatei.

b = block device (Blockorientiere Datei, Festplatte, Drucker,Maus etc.)

p = named pipe (FIFO)

l = symbolischer Link

s = socket Datei

Dateirechte:

Der nachfolgende 9er Block wird in 3 Bereiche mit je 3 zusammenhängenden Einträgen zusammengefasst.

r = read (lesen, ich kann lesen und die Datei ausdrucken)

w = write (schreiben, ich darf verändern und löschen)

x = executetable (ausführbar, kann z.B. dieses Programm oder Script starten, also ausführen)

Verzeichnisrechte:

Der nachfolgende 9er Block wird in 3 Bereiche mit je 3 zusammenhängenden Einträgen zusammengefasst.

r = read (lesen, ich kann lesen und die Datei ausdrucken)

w = write (schreiben, ich darf verändern und löschen)

x = man darf ins das Verzeichnis wechseln und auf dessen Inhalt zugreifen

Das x bei Verzeichnisrechten hat also eine grundlegend andere Bedeutung, denn ein Verzeichnis auszuführen macht wohl wenig Sinn.

Ist das x Recht auf ein Verzeichnis nicht gesetzt, kann ich zwar den Inhalt sehen, kann aber nicht darauf zugreifen, auch wenn die Dateirechte dazu vorhanden wären (dies als Anmerkung zu dem oben erwähnten Begriff „Vererbung“)

Also erhält man Informationen über den Inhalt der Dateien nur, wenn man neben dem Leserecht (r) auch das Recht (x) auf das jeweilige Verzeichnis hat.

o.g. Beispiel einmal aufgeschlüsselt:

```
d       rwx     r-x     r-x     root    root    4096    2010-10-23 14:41        Meine_Texte
|       |       |       |       |       |       |       |                       |
|       |       |       |       |       |       |       |                       Verzeichnisname
|       |       |       |       |       |       |       Datum und Uhrzeit der letzten Änderung
|       |       |       |       |       |       Dateigrösse
|       |       |       |       |       Gruppenzugehörigkeit des Eigentümers
|       |       |       |       Eigentümer des Verzeichnisses
|       |       |       Rechte anderen Benutzer (other)
|       |       Rechte der Gruppe (group)
|       Rechte des Eigentümers (owner)
Verzeichniseintrag
```

Wie man also sieht, handelt es sich um einen Verzeichniseintrag, gefolgt von den rwx Rechten des Eigentümers (dieser hat volle Verfügungsgewalt über diesen Eintrag), die Gruppe welcher der Eigentümer angehört hat die Rechte r-x (Gruppenmitglieder dürfen das Verzeichnis lesend einsehen und auch die darin enthaltenen Einträge lesen), andere Benutzer haben hierbei die gleichen Rechte r-x wie die Gruppenteilnehmer.

Das Verzeichnis gehört root und dieser gehört der Gruppe root an.

Bei normalen Dateien (-) und den anderen o.a. Typen von Device's (Gerätedateien etc.) sieht es bis auf die x Funktion (execute, ausführen) genauso aus.

Ein Benutzer kann auch die Rechte einer Datei oder eines Verzeichnisses nur so anpassen, wie es ihm durch die Rechtevergabe des jeweiligen Eintrags möglich ist. Er kann also keine Rechte hinzufügen, wenn er nicht bereits über diese Rechte auch verfügt.

Im Klartext heißt dies, er kann sich eigentlich nur in seinen Rechten weiter einschränken, diese jedoch nicht erweitern.

Die Ausnahme sollte natürlich sein, das root alle Dateirechte etc. verändert kann, solange er sich diese nicht selbst nimmt. ;-)

Wie wir sehen, sind die Rechte an dieser Stelle mit Buchstaben (rwx) belegt. Man kann also genau sehen wer was mit

dieser Datei machen darf und was nicht, geordnet nach Eigentümer, Gruppe und andere Benutzer.

Manchmal liest man aber etwas von 777 oder 644.

Was ist denn dies nun wird man sich fragen. Nun in der Regel das gleiche nur in einer anderen Schreibweise.

Man nennt dies „Oktalnotation“, bedeutet also, das die Kombinationen der Rechte mittels Oktalzahlen zusammengesetzt werden.

Oktalzahlen basieren auf der Basis von 8, nicht wie unser Dezimalsystem auf der Basis 10. Wie das geht werden wir nun nachfolgend erklären.

Oktalziffern für die Rechte:

0 = kein Recht

1 = x (execute, ausführen können)

2 = w (write, schreibrechte, löschrechte, Veränderungen vornehmen können)

3 = r (read, Leserechte)

Wir ordnen nun den einzelnen Rechten für Eigentümer (Besitzer), Gruppe und andere Benutzer nachfolgende Wertigkeiten zu:

Besitzer				**Gruppe**				**Andere Benutzer**		
r	w	x		r	w	x		r	w	x
400	200	100		40	20	10		4	2	1
	700		+		70		+		7	

und Zählen anschließend die Zahlen so zusammen das die gewünschten Zugriffsrechte hieraus gebildet werden können.

700 würde demnach bedeuten, das nur der Besitzer über alle Rechte verfügt, der Rest, Gruppe und andere Benutzer verfügen über keinerlei Rechte an diesem Eintrag.

777 würde bedeuten (wie im o.a. Beispiel), das alle Benutzer , alle Rechte an dieser Datei haben.

644 würde bedeuten, das der Besitzer Schreib und Leserechte besitzt, Gruppe und andere Benutzer dürften den Eintrag nur lesen. Keiner der Benutzer egal ob Besitzer oder Gruppenmitglied und „andere Benutzer“ können diese Datei ausführen.

Einfach, oder?

3. Rechte ändern und anpassen

Die genaue Syntax zu den einzelnen u.a. Befehlen kann man über ***Befehlsname --help*** oder eben ***man Befehlsname*** aus einer Konsole heraus erhalten.

Um normale Datei- und Verzeichnis Rechte zu ändern stehen 3 Befehle zur Verfügung:

1. **chown** (change owner, ändere den Besitzer eines Eintrags)
2. **chgrp** (change group, Rechte einer Gruppe zuweisen)
3. **chmod** (change mode, Änderungen an den Zugriffsrechten (oktale schreibweise)

chown

ändert den Besitzer eines Eintrags

Syntax:

chown *Besitzer Datei*
chown *Besitzer Datei1 Datei2 Datei3 .....*

Das ganze geht auch für alle Dateien und Recursiv, das heißt, es wird auch in den untergeordneten Verzeichnissen der Besitzereintrag angepasst bzw. verändert.

chown -R *Besitzer Datei*

Auch kann der Befehl zur Änderung des Benutzers und der Gruppe eingesetzt werden:

chown Benutzername:Gruppenname -R /var/www/Intranet

Hierbei würde der Benutzer und der Gruppenname des Verzeichnisses /var/Intranet und für alle enthaltenden Verzeichnisse und Dateien Recursiv geändert.

chgrp

ändert die Gruppenzugehörigkeit eines Eintrags

Syntax:

chgrp *Gruppenname Datei*

Ebenso gelten die gleichen Konventionen wie unter chown beschrieben (Recursion) auch für chgrp.

chmod

ändert die Rechte eines Eintrag

Syntax:

chmod *644 Datei*

Ebenso gelten die gleichen Konventionen wie unter chown beschrieben (Recursion) auch für chmod.

4. Besondere erweiterte Rechte und Funktionen

Wie heist es immer so schön, **Aufpassen**.

Die hier beschriebenen erweiterten Rechte sollten nur genutzt werden, wenn man weiß was man tut, bzw. tun muss um eine bestimmtes gewünschtes ***zukünftiges Verhalten*** innerhalb des Dateisystems zu erreichen.

Es werden nur die Grundzüge beschrieben was die Befehle eigentlich tun und wie man diese benutzt. Möchten Sie also genau wissen was passiert und wie diese Funktionen im Einzelnen greifen, werfen Sie einfach die Suchmaschine ihres Vertrauens an und informieren sich explizit über den jeweiligen Befehl oder die jeweilige Funktion und deren vorhandenen Parameter.

Hierzu zählen die Begriffe:

Sticky-Bit

umask

SUID-Bit

SGID-Bit

Sticky-Bit (T-Bit)

Aktuelle Bedeutung:

Das Sticky-Bit erlaubt nur dem User root und dem Besitzer einer Datei, in einem Verzeichnis mit gesetztem Sticky-Bit diese Datei zu löschen.

Anwendungsfall wäre zum Beispiel das Verzeichnis /tmp in dem zwar jeder Benutzer über alle Rechte verfügt aber das o.a. Verhalten erzwungen werden soll.

Die beiden möglichen Befehle würden demnach etwa so aussehen:

Syntax: chmod *u+t /tmp* oder chmod *1777 /tmp*

Bedeutung in der Vergangenheit:

In der Vergangenheit bedeutete dieses Bit, das das Programm nach dessen Beendung nicht aus dem Speicher entfernt wurde. Somit konnte man ein ständig benötigtes Programm ständig im Speicher halten ohne es entladen und wieder laden zu müssen. Schätze mal dies hat sich im Laufe der Zeit durch schnelle Festplatten und größere Hauptspeicher von selbst erledigt.

umask

Bei den Zugriffsbits wird die Sache etwas haarig. Linux geht als solches davon aus, dass jede neue Datei automatisch mit den Rechten (oktal 666) rw-rw-rw angelegt wird. Neu angelegte Programmdateien (neu kompilierte Programme) erhalten die Zuweisung 777 können also auch gleich von jedem ausgeführt werden.

Das diese Vorgehensweise aber nicht gerade einem sicheren System entsprechen würden, werden in der Datei /etc/profiles (oder /etc/bashrc) diese Rechte für neu angelegte Dateien auf (oktal 022) -r--r--r mittels umask festgelegt.

Diese Einstellung geschieht jedoch an dieser Stelle Systemweit. Möchte ein Benutzer seine neu angelegten Dateien jedoch mit anderen Rechten anlegen, so verwendet er hierzu die versteckte Datei im eigenen Homeverzeichnis (/home/Benutzername/.bashrc).

Prozesssteuerung - Zugriffsbits

SUID-Bit

Syntax: chmod u+s *Dateiname*

Das s.g. SUID Bit (Setuid-Bit) besagt, wenn es gesetzt ist, dass ein Programm mit den Rechten seines Besitzers gestartet wird.

Da in der Regel der Besitzer eines Programms der User root ist, bedeutet dies, das ein normaler Benutzer ein Programm bei gesetztem SUID Bit dieses mit Rootrechten starten kann.

Man beachte aber, das dies zu einem Sicherheitsrisiko führt. Man sollte also sehr Sparsam wenn besser gar nicht dieses Bit setzen wenn es sich vermeiden lässt.

Beim auflisten (ls -l) eines Inhaltsverzeichnisses wird ein gesetztes SUID Bit nicht im einem x sondern einem s innerhalb der ersten 3 Dateirechte des Besitzers angezeigt.

Damit ein Benutzer aber dieses Programm ausführen kann, muss natürlich bei seinen Rechten (Gruppe oder Dateirechte) auch ein x fürs Ausführen vorhanden sein.

SGID-Bit

Syntax: chmod g+s Dateiname

Setzt man dieses Bit (Setgid-Bit) auf ein Verzeichnis, so bedeutet dies, dass die neu erzeugten Dateien in dem Verzeichnis der Gruppe des jeweiligen Verzeichnisses gehören.

Letzte Aktualisierung 24.03.2011