



SV3: Linux Server

Dateiberechtigungen

Agenda

Dateiberechtigungen Kommandos

- chown
- chgrp
- chmod
- umask
- ACL (Access Control List)

Merkmale einer Datei

Eine Datei besteht aus einem Dateikopf, in dem bestimmte Merkmale der Datei eingetragen sind, und dem eigentlichen Dateiinhalt. Wenn wir uns mit **ls -l** eine Datei oder ein Directory ansehen, so bekommen wir einige dieser Merkmale angezeigt:

- **Dateityp** – (Directory, normale Datei oder Gerätedatei, symbolischer Link),
- **Zugriffsrechte** – (read, write, execute – Lese-, Schreib- und Ausführerlaubnis
- **für den Besitzer**, die Gruppe und die anderen Benutzer),
- **Referenzzähler** – (Anzahl der vergebenen Dateinamen),
- **Benutzername und Gruppe**
- **Größe in Bytes**
- **letztes Änderungsdatum und Name der Datei.**

Dateiberechtigungen

Beispiel: Anzeige der Dateien mit weiteren Dateiattributen (ls -lis)

| iNode-Nr. | Blockanzahl | Zugriffsrechte | Referenz-Zähler | Benutzer | Gruppe | Größe in Bytes | Datum der letzten Änderung | Dateiname |
|-----------|-------------|----------------|-----------------|----------|--------|----------------|----------------------------|-----------------------|
| 3932460 | 0 | -rw-r--r-- | 1 | sb | sb | 243 | Apr 16 | 1.txt |
| 3932461 | 0 | -rw-r--r-- | 1 | sb | sb | 1024 | Apr 16 | 2.sh |
| 3932462 | 0 | -rw-r--r-- | 1 | sb | sb | 8192 | Apr 16 | 3.txt |
| 3932463 | 0 | -rw-r--r-- | 1 | sb | sb | 1234 | Apr 16 | 4.text.txt |
| 3932257 | 4 | -rw-rw-r-- | 2 | sb | sb | 12 | Apr 17 | adressen |
| 3932513 | 4 | -rw-rw-r-- | 1 | sb | sb | 12 | Apr 17 | adressen_alphabetisch |
| 3932232 | 4 | drwxr-xr-x | 1 | sb | sb | 4096 | Apr 14 | Bilder |

Dateiberechtigungen

Beispiele: Zugriffsrechte

| Gerätedatei | | | |
|---|------------|------------|------------|
| Typ | Besitzer | Gruppe | Andere |
| c | rw- | --- | --- |
| Die Benutzer der gleichen Gruppe und andere dürfen auf diesem Gerät weder schreiben noch lesen. | | | |

| Verzeichnis | | | |
|--|------------|------------|------------|
| Typ | Besitzer | Gruppe | Andere |
| d | rwX | rwX | r-X |
| In dieses Verzeichnis dürfen alle mit cd wechseln (X) und die Dateien lesen (r). Neue Dateien anlegen und ändern (w) dürfen nur der Besitzer und die Benutzer der gleichen Gruppe. | | | |

Dateiberechtigungen

Beispiele: Zugriffsrechte

| Dateien | | | |
|---|------------|------------|------------|
| Typ | Besitzer | Gruppe | Andere |
| - | rw- | rw- | r-- |
| Diese Datei darf von allen gelesen werden. Verändert werden darf sie nur vom Besitzer oder von Benutzern der gleichen Gruppe. | | | |
| - | rwX | rwX | r-X |
| Diese Datei darf von allen gelesen und ausgeführt werden. Verändern dürfen nur der Besitzer und die Benutzer der gleichen Gruppe. | | | |

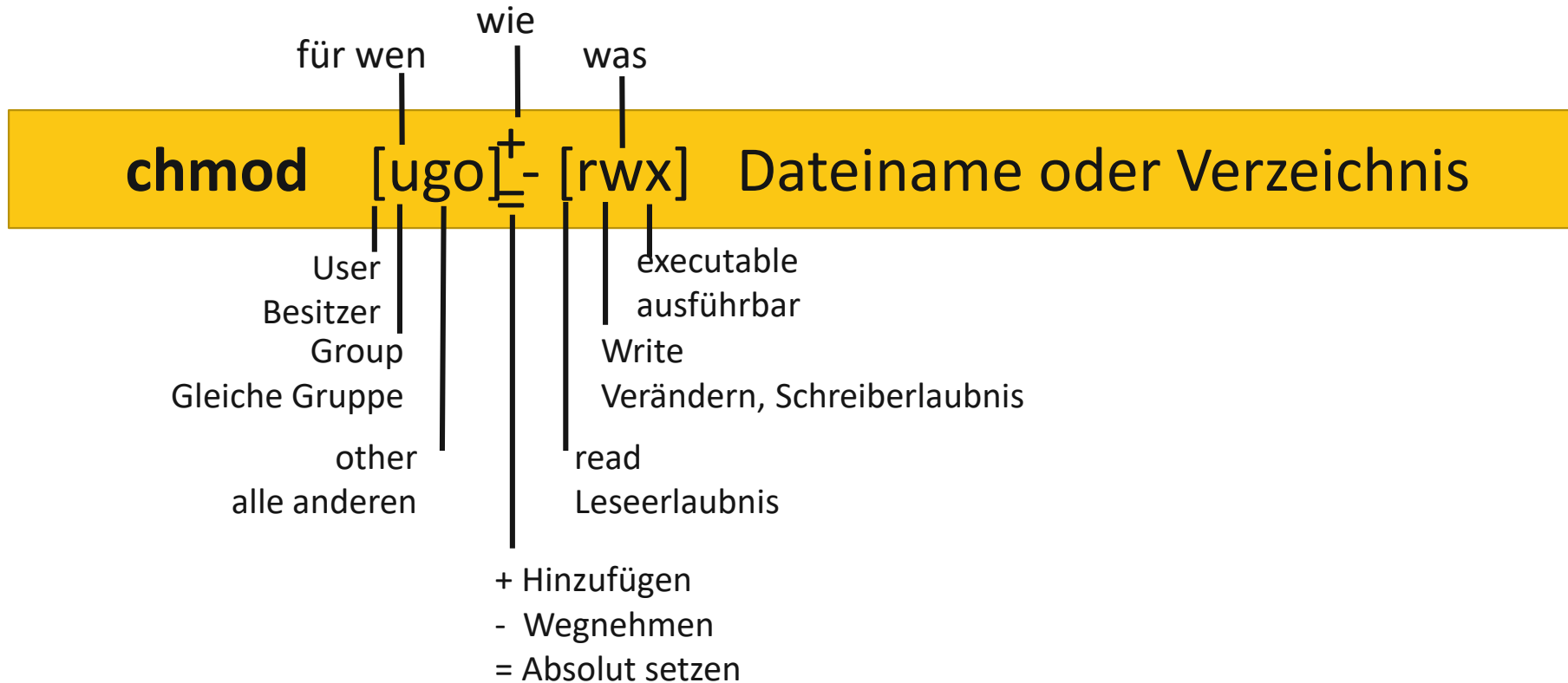
Dateiberechtigungen

chmod

Nur wenn Sie Besitzer einer Datei oder eines Directories sind, können Sie die Zugriffsrechte ändern. Hierfür wird das Kommando **chmod** (change modus) verwendet.

- Dieses Kommando können Sie unterschiedlich ausführen:

Dateiberechtigungen



chmod – kommando, um Zugriffsrechte zu ändern
(symbolische Angabe)

Dateiberechtigungen

Mit dem **chmod**-Kommando verändern Sie die Zugriffsrechte von Dateien, wobei Sie angeben müssen:

| | |
|----------------|---|
| Für wen | Besitzer (user) Benutzer der gleichen Gruppe (group) Alle anderen (other) Wird nichts angegeben, gilt die Änderung für alle! |
| wie | Sollen die Zugriffsrechte Hinzugefügt (+) werden? Oder sollen die Zugriffsrechte entzogen (-) werden? Eine von beiden Optionen muss angegeben werden! |
| was | Welche Rechte sollen geändert werden? Lesen (read) Schreiben (write) Ausführen (execute) Eine von beiden Optionen muss angegeben werden! |

Dateiberechtigungen

Sollen beispielsweise alle Mitglieder die Erlaubnis zum Schreiben bekommen, so lautet das Kommando:

```
chmod g+w Dateiname
```

Dateiberechtigungen

Beispiel: Ändern der Zugriffsrechte mit symbolischer Angabe –

chmod ugo+rwX

Sie sehen an dem Beispiel, dass Kombinationen von go und wx erlaubt sind. Wichtig ist, dass bei der Angabe, für wen, wie und was geändert werden soll, keine Leerzeichen zwischen den einzelnen Symbolen stehen dürfen.

```
sb@ub:~/test$ ls -la
insgesamt 8
drwxr-xr-x  2 sb sb 4096 Apr 20 15:34 .
drwxr-xr-x 20 sb sb 4096 Apr 20 15:34 ..
-rw-r--r--  1 sb sb   0 Apr 20 15:34 inhalt
-rw-r--r--  1 sb sb   0 Apr 20 15:34 inhalt1
-rw-r--r--  1 sb sb   0 Apr 20 15:34 inhalt2
sb@ub:~/test$
sb@ub:~/test$ chmod go+wx inh*
sb@ub:~/test$
sb@ub:~/test$ ls -la
insgesamt 8
drwxr-xr-x  2 sb sb 4096 Apr 20 15:34 .
drwxr-xr-x 20 sb sb 4096 Apr 20 15:34 ..
-rw-rwxrwx  1 sb sb   0 Apr 20 15:34 inhalt
-rw-rwxrwx  1 sb sb   0 Apr 20 15:34 inhalt1
-rw-rwxrwx  1 sb sb   0 Apr 20 15:34 inhalt2
sb@ub:~/test$
```

Dateiberechtigungen

Ausführen von chmod mit Zahlenwerten (Oktalzahl)

Für die einzelnen Rechte werden folgende Werte vergeben:

chmod [777] Dateiname oder Verzeichnis

User
Besitzer
Group
Gleiche Gruppe
other
alle anderen

| | | | |
|---|---------|-----------|---|
| r | read | Lesen | 4 |
| w | write | Schreiben | 2 |
| x | execute | Ausführen | 1 |

Werden in einer Datei alle Rechte gesetzt, ist der Wert 777

Dateiberechtigungen

Beispiel: Alle Rechte einer Datei sind gesetzt

Sollen die Rechte verändert werden, so wird nach den drei Unterteilungen Besitzer Gruppe andere die jeweilige Summe der zugewiesenen Rechte eingesetzt.

- Darf z.B. der Besitzer lesen, schreiben und ausführen, die Gruppe nur lesen und ausführen, und die anderen nur lesen, so ergibt sich folgende Berechnung der Oktalzahl

| Zugriffsrechte (Modus) eingeteilt nach: | | | | | | | | |
|---|-------|---------|----------------|-------|---------|----------------|-------|---------|
| Besitzer (user) | | | Gruppe (group) | | | Andere (other) | | |
| read | write | execute | read | write | execute | read | write | execute |
| 4 | 2 | 1 | 4 | 0 | 1 | 4 | 0 | 0 |
| 7 | | | 5 | | | 4 | | |

Dateiberechtigungen

Beispiel: Ändern der Zugriffsrechte mit Oktalzahl

Die Oktalzahl wurde folgendermaßen berechnet:

| Besitzer | Gruppe | Andere |
|----------|--------|--------|
| rwX | r-X | --- |
| 4+2+1 | 4+0+1 | 0+0+0 |
| = 7 | = 5 | = 0 |

```
sb@ub:~/test$ ls -l
insgesamt 0
-rw-rwxrwx 1 sb sb 0 Apr 20 15:34 inhalt
-rw-rwxrwx 1 sb sb 0 Apr 20 15:34 inhalt1
-rw-rwxrwx 1 sb sb 0 Apr 20 15:34 inhalt2
sb@ub:~/test$
sb@ub:~/test$ chmod 750 inhalt*
sb@ub:~/test$
sb@ub:~/test$ ls -l
insgesamt 0
-rwxr-x--- 1 sb sb 0 Apr 20 15:34 inhalt
-rwxr-x--- 1 sb sb 0 Apr 20 15:34 inhalt1
-rwxr-x--- 1 sb sb 0 Apr 20 15:34 inhalt2
sb@ub:~/test$
```

Würde der Besitzer der Datei einem anderen Benutzer eine seiner Dateien in sein Directory kopieren, bleibt sie Besitzer der kopierten Datei.

- Dabei spielt es keine Rolle, von wo sie den Kopierauftrag startet.
- Das heißt: Auch wenn der Besitzer der Datei zuvor in das Directory des anderen Benutzers wechselt und dann kopiert, bleibt er trotzdem alleiniger Besitzer der kopierten Datei

Dateiberechtigungen

chown steht für change owner und erlaubt das Ändern des Eigentümer-Benutzers und/oder der Eigentümer-Gruppe von Dateien.

Dies funktioniert jedoch nur bei Dateisystemen, welche die UNIX-Dateirechte unterstützen (z.B. ext2, ext3 und ext4) Bei FAT ist dies grundsätzlich nicht der Fall, und bei NTFS erfordert dies die Mount-Option permissions (ist standardmäßig nicht gesetzt).

- Die allgemeine Syntax ist wie folgt:

chown [-R] Besitzer:Gruppe Dateiname(n) oder Verzeichnis(se)

Die Option -R ist optional und erlaubt es, die Unterverzeichnisse und deren Dateien mit einzubeziehen

Dateiberechtigungen

Um die kopierte Datei auf tux umzuschreiben und ihm die Besitzrechte zu übergeben, muss der Administrator Folgendes eingeben:

```
sb@ub:~/test$ ls -l
insgesamt 0
-rwxr-x--- 1 tux tux 0 Apr 20 15:34 inhalt
-rwxr-x--- 1 sb sb 0 Apr 20 15:34 inhalt1
-rwxr-x--- 1 sb sb 0 Apr 20 15:34 inhalt2
sb@ub:~/test$
sb@ub:~/test$ sudo chown tux:tux inhalt
sb@ub:~/test$
sb@ub:~/test$ ls -l
insgesamt 0
-rwxr-x--- 1 tux tux 0 Apr 20 15:34 inhalt
-rwxr-x--- 1 sb sb 0 Apr 20 15:34 inhalt1
-rwxr-x--- 1 sb sb 0 Apr 20 15:34 inhalt2
sb@ub:~/test$
```

chown kennt dabei unter anderem folgenden Optionen:

| Optionen von chown | |
|-----------------------------------|---|
| Option | Beschreibung |
| -c oder --changes | Es werden (nur) die Dateien angezeigt, deren Besitzer tatsächlich verändert wird. |
| -f oder --force | Fehlermeldungen wegen fehlgeschlagener Änderungsversuche werden unterdrückt |
| -v oder --verbose | Alle Aktion werden angezeigt |
| -R oder --recursive | Der Besitzer aller Dateien in den Unterverzeichnissen wird ebenfalls geändert |

Dateiberechtigungen

Nachfolgend sind alle Kombinationsmöglichkeiten von Besitzer und Gruppe aufgeführt, die von chown akzeptiert werden:

| Kombinationsmöglichkeiten von Besitzer und Gruppe | |
|---|--|
| Kombination | Bedeutung |
| besitzer_name:gruppen_name | Benutzer und Gruppe werden auf einen Schlag gesetzt. |
| :gruppen_name | Die Gruppe wird gesetzt, wohingegen der Besitzer unverändert bleibt. |
| besitzer_name: | Der Besitzer wird auf besitzer_name - und die Gruppe auf die Standardgruppe des eingeloggten Benutzers gesetzt. |
| besitzer_name | Ausschließlich der Besitzer wird gesetzt. |

chgrp

steht für change group und ändert die Gruppenzugehörigkeit von Dateien und Ordnern. Man kann diesen Befehl nur auf eine Datei anwenden, wenn man der Eigentümer oder der Superuser ist. Es stehen nur die Gruppen denen man selber angehört zur Verfügung.

- Die allgemeine Syntax ist wie folgt:

chgrp [Option(en)] Gruppe Dateiname(n) oder Verzeichnis(se)

Dateiberechtigungen

chgrp kennt dabei unter anderem folgende Optionen:

| Optionen von chgrp | |
|-----------------------------------|--|
| Option | Beschreibung |
| -c oder --changes | Diese Option zeigt die Dateien an, die geändert werden. |
| -f oder --force | Fehlermeldungen werden unterdrückt. |
| -v oder --verbose | Diese Option zeigt alles was chgrp macht. |
| -R oder --recursive | Diese Option ermöglicht das rekursive Ändern von Verzeichnissen. |

Dateiberechtigungen

Ein User ist in der Gruppe `musik` und ändert die Gruppenzuordnung des Verzeichnisses `/opt/musik/` und seinem kompletten Inhalt zu `musik`:

```
chgrp -c -R musik /opt/musik/
```

Sonderrechte

Für besondere Anwendungen gibt es zusätzlich noch besondere Dateirechte. Der Einsatz dieser ist nur dann ratsam, wenn man genau weiß, was man tut, da dies unter Umständen zu Sicherheitsproblemen führen kann.

| Sonderrechte | | |
|--------------|--------------------------|--|
| Zeichen | Bedeutung | Beschreibung |
| s | Set-UID-Recht (SUID-Bit) | Das Set-UID-Recht („Set User ID“ bzw. „Setze Benutzerkennung“) sorgt bei einer Datei mit Ausführungsrechten dafür, dass dieses Programm immer mit den Rechten des Dateibesitzers läuft. Bei Ordnern ist dieses Bit ohne Bedeutung. |
| s (S) | Set-GID-Recht (SUID-Bit) | Das Set-GID-Recht („Set Group ID“ bzw. „Setze Gruppenkennung“) sorgt bei einer Datei mit Ausführungsrechten dafür, dass dieses Programm immer mit den Rechten der Dateigruppe läuft. Bei einem Ordner sorgt es dafür, dass die Gruppe an Unterordner und Dateien vererbt wird, die in diesem Ordner neu erstellt werden. |
| t (T) | Sticky-Bit | Das Sticky-Bit („Klebrig“) hat auf modernen Systemen nur noch eine einzige Funktion: Wird es auf einen Ordner angewandt, so können darin erstellte Dateien oder Verzeichnisse nur vom Dateibesitzer gelöscht oder umbenannt werden. Verwendet wird dies z.B. für /tmp. |

Dateiberechtigungen

Experten-Info:

Alle Rechte werden durch entsprechend gesetzter Bits repräsentiert. Mit der Wertigkeit der gewünschten Bits kann dann entsprechend der Oktalwert errechnet werden.

| Recht | Wert |
|-----------|------|
| Lesen | 4 |
| Schreiben | 2 |
| Ausführen | 1 |

| Recht | Wert |
|---------|------|
| Set-UID | 4 |
| Set-GID | 2 |
| Sticky | 1 |

Sonderrechte

Die Symbole für die Sonderrechte erscheinen an der dritten Stelle der Zugriffsrechte, die normalerweise dem Zeichen x (für executable) vorbehalten ist, und ersetzen ggf. dieses.

- Die Set-UID/GID-Rechte werden anstelle des x für den Besitzer bzw. die Gruppe angezeigt, das Sticky-Bit anstelle des x für andere.
- Wenn das entsprechende Ausführrecht gesetzt ist, wird ein Kleinbuchstabe verwendet, ansonsten ein Großbuchstabe.

Dateiberechtigungen

Achtung!

Vor allem das "Set-UID-Recht" sollte nur mit äußerster Vorsicht angewandt werden!

Besonders gefährlich ist es, das SUID-Bit bei Dateien zu setzen, die Besitz von root sind, denn dann kann jeder diese mit Administrator-Rechten starten. Schwachstellen im jeweiligen Programm können dann dazu ausgenutzt werden, vollständigen Root-Zugriff auf das gesamte System zu bekommen. Nur bei wenigen Programmen, die darauf ausgelegt sind, ist dieses Bit gesetzt (z.B. sudo oder su).

Dateiberechtigungen

Standard-Einstellung und Maskierung

In jedem Linux-System gibt es Standardwerte für die Zugriffsrechte bei neu erstellten Ordnern und Dateien. Diese können in einem Terminal mit dem Befehl **umask** abgefragt werden.

| Befehl | Ausgabe | Beschreibung |
|-----------------|---------------------------|---|
| umask | 0002 | Darstellung in oktaler Form. Die erste Ziffer repräsentiert hierbei das Sonderrecht, die anderen drei stehen jeweils für „Eigentümer“, „Gruppe“ und „alle anderen“. |
| umask -s | u=rwx, g=rwx, o=rx | Darstellung in symbolischer Form. Das „u=“ steht für „user“ (Eigentümer), das „g=“ für „groups“ (Gruppe) und das „o=“ für „other“ (alle anderen). Hinter dem Gleichheitszeichen stehen die jeweiligen Rechte. |

In der oktalen Form (umask ohne Parameter)

werden diejenigen Rechte angezeigt, die maskiert sind, d.h. die nicht gesetzt werden. In der symbolischen Form werden hingegen diejenigen Rechte angezeigt, die nach der Maskierung bestehen bleiben.

■ Der hier Standardwert 0002 bedeutet:

| Rechte | Beschreibung |
|---------|--|
| S = --- | Keine Sonderrechte gesetzt |
| U = rwx | Eigentümer darf lesen, schreiben und ausführen |
| G = rwx | Die Gruppe darf lesen, schreiben und ausführen |
| O = r-x | Alle anderen dürfen nur lesen und ausführen |

Dateiberechtigungen

- Möchten Sie den Wert ändern, so geben Sie diesen den gewünschten Wert oder die Rechte als Parameter:

```
sb@ub:~$ umask -S
u=rwx,g=rwx,o=rx
sb@ub:~$
sb@ub:~$ umask g-wx
sb@ub:~$ # oder
sb@ub:~$ umask 0032
sb@ub:~$
sb@ub:~$ umask -S
u=rwx,g=r,o=rx
sb@ub:~$
sb@ub:~$ # Der Gruppe (g) wird das Recht Schreiben und Ausführen entzogen. Umgekehrt können Sie die Rechte auch wieder
          vergeben. Gleichzeitig wird allen anderen (o), dass Lese und Ausführrecht entzogen.
sb@ub:~$
sb@ub:~$ umask g+wx,o-rx
sb@ub:~$ # oder
sb@ub:~$ umask 0007
sb@ub:~$
sb@ub:~$ umask -S
u=rwx,g=rwx,o=
sb@ub:~$
```

Die so eingegebenen Werte von **umask** gelten nur für die betreffende Sitzung.

Dateiberechtigungen

Die systemweite Einstellung von umask, die früher durch einen Eintrag in **/etc/profile** vorgenommen wurde, wird nun vom PAM-Modul **pam_umask** über die Datei **/etc/login.defs** erledigt.

- Für einzelne Benutzer können abweichende Einstellungen durch einen Eintrag in **~/.profile** festgelegt werden.
- Änderungen werden jeweils erst nach einem Neustart bzw. einer Neuansmeldung wirksam.

Dateiberechtigungen

Berechnung der Maske

Die Maske ist der invertierte Wert der gewünschten Zugriffsrechte.

- Die einfachste Möglichkeit die Maske zu ermitteln, besteht darin die gewünschten oktalen Werte jeweils von 7 abzuziehen

| Differenz zu 777 | | | |
|------------------|---|---|---|
| Ausgangswert | 7 | 7 | 7 |
| Maske | 1 | 3 | 7 |
| Effektive Rechte | 6 | 4 | 0 |

Dateiberechtigungen

Berechnung umask am Beispiel:

| Zugriffsrechte | | | | | | | | | |
|----------------|----------|---|---|--------|---|---|--------|---|---|
| | Benutzer | | | Gruppe | | | Andere | | |
| symbolisch | r | w | - | r | - | - | - | - | - |
| oktal einzeln | 4 | 2 | 0 | 4 | 0 | 0 | 0 | 0 | 0 |
| oktal | 6 | | | 4 | | | 0 | | |
| Maske | | | | | | | | | |
| | Benutzer | | | Gruppe | | | Andere | | |
| oktal | 1 | | | 3 | | | 7 | | |
| oktal einzeln | 0 | 0 | 1 | 0 | 2 | 1 | 4 | 2 | 1 |
| symbolisch | - | - | x | - | w | x | r | w | x |

Access Control Lists (kurz ACL)

Mit Hilfe von Access Control Lists (kurz ACL) ist es möglich, einzelnen Nutzern (oder auch Gruppen) gezielt Rechte an einzelnen Dateien zu geben oder zu entziehen.


- ACLs ergänzen damit die normale Rechteverwaltung von Linux. Außerdem kann man mit ACLs die einheitliche Vergabe von Rechten für neu angelegte Dateien innerhalb eines Verzeichnisbaumes erzwingen.
- Dies kann insbesondere auf größeren Mehrbenutzersystemen nützlich sein. Auf "normalen" Desktop-Systemen mit einem oder zwei Nutzern ist der Einsatz von ACLs üblicherweise nicht sinnvoll.
- Zudem gibt es viele Shellbefehle, die ACL ignorieren und Probleme verursachen können.

Dateiberechtigungen

Aktivieren im Dateisystem

- Bei den Dateisystemen JFS und XFS können ACLs standardmäßig gesetzt werden.
- Bei den unter Linux üblichen Dateisystemen ext3, ext4 und Reiserfs müssen ACLs erst explizit aktiviert werden. Dies geschieht durch die Option `-o acl` beim Einbinden der Partition oder direkt in `/etc/fstab`.

```
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda2 during installation
UUID=c20aef0f-47bf-4b1e-82e1-db6b039905f9 / ext4 errors=remount-ro,acl 0 1
# /boot/efi was on /dev/sda1 during installation
UUID=4E89-7C53 /boot/efi vfat umask=0077 0 1
/swapfile none swap sw 0 0
```



Seit Ubuntu 12.04 gehört `acl` bei den Dateisystemen ext3 und ext4 zu den Default-Optionen und braucht deshalb auch bei diesen nicht mehr explizit aktiviert zu werden.

Bearbeiten von ACLs

Die ACL-Verwaltung läuft über zwei Programme:

setfac

dient zum Setzen und Löschen von ACLs

getfac

dient zum Auslesen von ACL

Wie bei den klassischen UNIX Rechten, können auch ACLs für eine Datei nur vom Besitzer der jeweiligen Datei gesetzt werden, nicht von anderen Nutzern. Darüber hinaus funktionieren die Befehle ls und chmod weiterhin, wenn auch leicht verändert.

Dateiberechtigungen

- Die klassischen Unixrechte erlauben nur Unterscheidung der Zugriffsrechte für den Besitzer, die besitzende Gruppe und Anderen ("other").
- Will man genau einer weiteren Person Rechte auf eine Datei erteilen, muss man eine Gruppe anlegen, in der genau die Benutzer enthalten sind, die die Rechte bekommen sollen.
- Für jede gewünschte Kombination von Benutzern benötigt man also eine eigene Gruppe, was unpraktikabel ist.

Aufbau einer ACL

ACLs lösen das Problem, indem man beliebigen weiteren Personen und Gruppen die Zugriffsrechte (lesen, schreiben, ausführen bzw. rwx) erteilen kann. ACLs erlauben die Rechtevergabe an folgende Parteien:

- genau ein Besitzer, benannte Benutzer
- genau eine besitzende Gruppe, benannte Gruppe
- Andere

Der Besitzer, die besitzende Gruppe und Andere sind die gleichen wie bei den klassischen Unixrechten. Neu sind keine, eine oder mehrere benannte Benutzer und benannte Gruppen.

Setzen und Löschen von ACLs

Setzen von Rechten für Benutzer und Gruppen

```
setfacl -m u:Benutzer:-,g:Gruppe:RECHTE,... DATEI ...
```

Man beachte, dass mehrere Rechtevergaben durch Kommas getrennt aufgelistet werden können. Entfernen einzelner Einträge in der ACL:

```
setfacl -x u:BENUTZER,g:GRUPPE,... DATEI ...
```

Dateiberechtigungen

- Beim Löschen einzelner Einträge einer ACL wird die Maske neu berechnet.
- Dabei können unbeabsichtigt benannte Benutzer und Gruppen Rechte zugeteilt bekommen. Das Neuberechnen der Maske kann man mit Option **-n** verhindern.
- Entfernen der gesamten ACL, so dass nur die klassischen Unixrechte zurückbleiben:

```
setfacl -b DATEI ...
```

Dateiberechtigungen

Beispiel

Der Nutzer **Anton** erstellt Dateien und lässt seine Kollegen in der Gruppe **Schreiber** lesen. Sie hat keine ACL, aber man kann die Zugriffsrechte in ACL-Form abrufen:

```
anton@ub:~$ ls -l roman.txt
-rw-r--r-- 1 anton schreiber 0 Apr 20 11:22 roman.txt
anton@ub:~$ getfacl roman.txt
# file: roman.txt
# owner: anton
# group: schreiber
user::rw-
group::r--
other::r--
```

Da (noch) keine ACLs gesetzt sind, entspricht die Ausgabe im Prinzip der oben gezeigten von `ls -l`, lediglich mit einer anderen Darstellung. Anton möchte verhindern, dass sein Chef, der ebenfalls in der Gruppe `schreiber` ist, die Datei lesen kann. Gleichzeitig möchte er den Lektoren die Möglichkeit geben, seine Datei zu korrigieren.

Dateiberechtigungen

Prioritäten

Welcher Eintrag für die Zugriffsrechte entscheidend ist, bestimmen folgende Regeln:

- Die ACL wird von oben nach unten abgearbeitet.
- Die erste zutreffende Regel gilt.

Beispiel:

```
# file:    roman .txt
# owner: anton
# group:  schreiber
user: : rw-
user: chef: ---
user : anton: r--
group: : r--
group: lektoren : rw-
mask: : rw-
other: : r--
```

Anton ist der Besitzer der Datei. Für ihn gelten die Rechte des Besitzers user::rw-. Der Eintrag user:anton:r-- folgt später und wird daher ignoriert.

Der Chef sei in der besitzenden Gruppe schreiber, welche lesen darf (group::r--). Trotzdem hat der Chef überhaupt keinen Zugriff, weil er weiter oben als benannter Benutzer ohne Rechte (user:chef:---) eingetragen ist.

Dateiberechtigungen

Maske

Die Maske legt die maximalen Rechte fest, die ein anderer Benutzer als der Besitzer oder eine Gruppe haben kann. Das heißt, wenn ein Eintrag eine Gruppe die Lese- und Schreibrechte einräumt, die Maske aber nur Leserechte vorsieht, dann hat diese Gruppe auch nur Leserechte. Ein Beispiel:

```
# file      roman .txt
# owner: anton
# group: schreiber
user: : rw-
user : chef : rw-                #effective : r--
group: : rwx                    #effective : r-x
group: lektoren: -wx            #effective : --x
mask: : r-x
other : : ---
```

Dateiberechtigungen

Der Befehl **getfacl** gibt die effektiven Rechte aus, wenn die Rechte durch die Maske eingeschränkt wurden. Die Maske kann auf 4 verschiedene Weisen berechnet bzw. gesetzt werden:

| Befehl | Option | Bedeutung |
|---------|-----------------------------|-----------------------|
| setfacl | -m RECHTELISTE ... | Automatisch berechnen |
| setfacl | -n -m RECHTELISTE ... | Nicht verändern |
| setfacl | -m m::MASKE,RECHTELISTE ... | Manuell setzen |
| chmod | g=MASKE | Manuell setzen |

Wenn nicht anders angegeben, wird die ACL beim Modifizieren der ACL automatisch so angepasst, dass sie die Rechte der neuen Einträge nicht einschränkt.

ACL entfernen

Wenn die ACL ganz entfernt wird, dann werden die Gruppenrechte gleich den effektiven Rechte der besitzenden Gruppe gesetzt, also der Maske geschnitten mit den Rechten der besitzenden Gruppe.

Zum Verständnis:

- Der Grund für diese Konstruktion ist Kompatibilität von alten Programmen.
- Manche Programme müssen kurz alle fremden Zugriffe auf einzelne Dateien verhindern, und benutzen dafür **chmod**.



**VIELEN DANK
FÜR IHRE
AUFMERKSAMKEIT!**