1.10 Anwendung/ACLs

1.10.1 Zugriffs-ACLs

Für die meisten Fälle reichen die bisherigen Standard-Unix-Berechtigungen aus. In komplizierten Fällen (z.B. zwei Gruppen + 3 Nutzer sollen Zugriff auf eine Datei bekommen) gestaltet sich die Lösung aber etwas kompliziert (Einrichtung neuer Gruppen usw.).

Für diese Fälle bietet sich die Verwendung so genannter Zugriffs-ACLs (access control lists) an. An jedes Dateisystemobjekt wird dabei eine endlich lange Liste von Zuordnungen geheftet:

- Nutzer1:Rechte
- Nutzer2:Rechte
- Gruppe1:Rechte usw.

Dieses System der Zugriffs-ACLs muss allerdings kompatibel zu den Standard-Unix-Berechtigungen sein. Dazu gibt es einen Posix-Norm-Vorschlag, der in Linux und vielen anderen Unix-Systemen übernommen wurde.

1.10.2 Minimale Zugriffs-ACLs

Die bisherigen Standard-Unix-Rechte heissen ab jetzt *Minimale Zugriffs-ACLs*. Der Zugriff auf die ACL erfordert ab jetzt zwei neue Befehle, nämlich `getfacl` (Lesen der ACL einer Datei) und `setfacl` (Schreiben der ACL einer Datei):

Terminal

```
schueler@debian964:~$ getfacl notiz.txt
# file: notiz.txt
# owner: schueler
# group: schule
user::rw-
group::r--
other::r--
```

Die Rechte des Eigentümers verbergen sich also in der Zeile, die mit `user::` beginnt, die Rechte der Gruppe der Datei in der Zeile mit `group::`, die Rechte der Anderen in der Zeile mit `other::` (Abbildung 1). Weiter heißen Eigentümer, Gruppe und Andere jetzt *Benutzerklassen*.

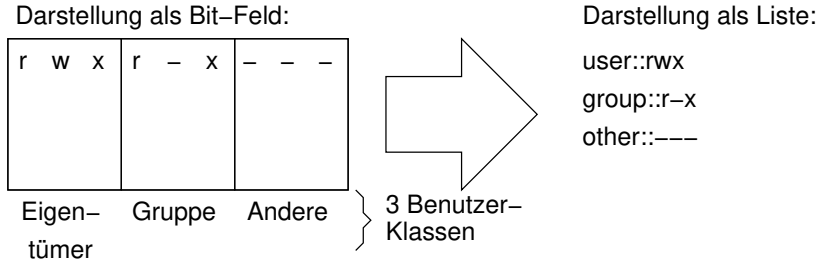

Abbildung 1: Minimale Zugriffs-ACLs

1.10.3 Erweiterte Zugriffs-ACLs

Nun folgt die Erweiterung auf wirkliche ACLs:

- Benutzerklasse Eigentümer: keine Änderung nötig (meist maximale Rechte)
- Benutzerklasse Andere: keine Änderung nötig (meist minimale Rechte)

- Benutzerklasse Gruppe: Hier wird erstens erweitert um ein Element *Maske*, welches wieder 3 Bit für die Rechte enthalten kann. Ab jetzt heißt die ACL eine Erweiterte Zugriffs-ACL.
- Ab jetzt können in der Benutzerklasse Gruppe beliebige weitere Listenelemente eingefügt werden:
  a) Rechte für bestimmte explizit benannte Benutzer *named users*),
  b) Rechte für bestimmte explizit benannte Gruppen (*named groups*).

Die Nennung eines *named users* erfolgt nach dem Muster `user:willi:`, die Nennung einer *named group* wie bei `group:is2a:`. Ein solcher Aufruf von `setfacl` lautet dann:

Terminal

```
schueler@debian964:~$ setfacl --modify user:willi:r,group:audio:rw \
   notiz.txt
schueler@debian964:~$ getfacl notiz.txt
# file: notiz.txt
# owner: schueler
# group: schule
user::rw-
user:willi:r--
group::r--
group:audio:rw-
mask::rw-
other::r--
```

Die Benutzerklasse Gruppe enthält nun also wie bisher die Standardgruppe, dazu (falls gewünscht) Named Users und (falls gewünscht) Named Groups (Abbildung 2). Die Auswertungsrei-

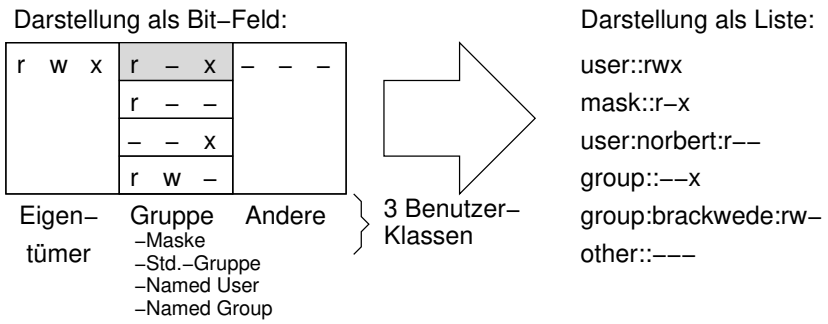

Abbildung 2: Erweiterte Zugriffs-ACLs

henfolge ist dann:

a) Eigentümer,
b) Named Users,
c) Gruppe und Named Groups,
d) Andere.

In dieser Reihenfolge fehlt das Element Maske (angegeben in der Zeile, die mit `mask::` beginnt). Die Maske bestimmt die maximal möglichen Rechte, also die Obergrenze für *alle* Mitglieder der Benutzerklasse Gruppe.

Die effektiven Rechte der *named group* `audio` ergibt sich so als die UND-Verknüpfung aus `group:audio:`-Rechten und `mask::`-Rechten. Schränkt man die Rechte der Maske ein, so wird auch `audio` eingeschränkt. Die Einschränkung ist am Ende der `group:audio:`-Zeile vermerkt:

Terminal

```
schueler@debian964:~$ setfacl --modify mask:rx notiz.txt
schueler@debian964:~$ getfacl notiz.txt
# file: notiz.txt
# owner: schueler
# group: schule
user::rw-
user:willi:r--
group::r--
group:audio:rw-                 #effective:r--
mask::r-x
other::r--
```

Bei Erweiterter Zugriffs-ACL übernimmt die Maske also die Vorherrschaft in der Benutzerklasse Gruppe und damit quasi die Rolle von `group::`. Daher verändert der `chmod`-Befehl bei Vorhandensein der Erweiterten Zugriffs-ACL nur die Maske, nicht aber die `group::`-Bits.

Mit der Maske kann so die Übersicht behalten werden über die Rechte, die an irgendjemand aus der Benutzerklasse Gruppe vergeben werden können.

1.10.4 Befehle für Zugriffs-ACLs

Der `setfacl`-Befehl hat folgende wichtige Optionen:

- `-m`, `--modify`: Modifizieren (wird fast immer gebraucht), Vorsicht: Maske wird gesetzt, und zwar auf Maximum aller Mitglieder der Benutzerklasse Gruppe!
- `-n`, `--no-mask`: Maske wird *nicht* gesetzt; andernfalls würde die Maske beim `setfacl`-Befehl immer auf das Maximum der Rechte aller Mitglieder in der Benutzerklasse Gruppe gesetzt.
- `-b`, `--remove-all`: löschen (neu anfangen)

Beim Kopieren von ACLs muss man aufpassen:

- `ls -l` zeigt mit einem + nach den Berechtigungsbits an, dass eine Erweiterte ACL vorliegt.
- `cp` ist unproblematisch, die Option -p kopiert nicht nur die Standard-Unix-Berechtigungen, sondern auch ACLs.
- Das Backup-Programm `tar` kennt noch keine ACLs. Stattdessen muss `star` benutzt werden.
- Das Backup-Programm `cpio` kennt bisher keine ACLs.
- `getfacl --recursive > datei.txt` schreibt alle ACLs in diesem Verzeichnis und darunter (rekursiv: auch Unterverzeichnisse) in eine Textdatei aus. Im Notfall kann deren Inhalt mit der Befehlszeile `setfacl --restore=datei.txt` wieder restauriert werden.

1.10.5 `umask` und Vorgabe-ACLs

Mit dem Befehl `umask` kann man default-Rechte für neu angelegte Dateien und Verzeichnisse festlegen. Nach der Eingabe von `umask 0027` kann man beobachten:

- Neu angelegte Verzeichnisse erhalten die Rechte 0750, da vom Maximum (0777) das r-Bit der Gruppe (2) und die rwx-Bits für die Anderen (7) abgezogen werden.
- Neu angelegte Dateien erhalten die Rechte 0640, da vom Maximum (0666) das r-Bit der Gruppe (2) und die rwx-Bits für die Anderen (7) auf null gesetzt werden (das x-Bit ist schon auf null, wird also nicht geändert).

`umask` ist also ein Befehl, der nicht notwendig ist, sondern nur den Komfort des Systems erhöht. Bei ACLs reicht dieser Mechanismus nicht mehr aus, weil in der Oktalzahl nicht genug Information gespeichert wird. Als hervorragenden Ersatz gibt es jedoch die sogenannten Vorgabe-ACLs (*default ACLs*) für Verzeichnisse.

Zu den bisherigen Einträgen in der ACL werden neue Einträge hinzugefügt, die zusätzlich das Präfix (die Vorsilbe) `default:` tragen, also z.B.:

- `default:user::rwx`,
- `default:group:is2a:rw`,
- `default:user:mueller:rw` oder
- `default:other::-`

Dazu kann man wie bisher den `setfacl`-Befehl verwenden (hier für eine Minimale ACL, also ohne Maske):

Terminal

```
schueler@debian964:~$ mkdir briefe
schueler@debian964:~$ setfacl --modify default:other:- briefe
schueler@debian964:~$ getfacl briefe
# file: briefe
# owner: schueler
# group: schule
user::rwx
group::r-x
other::r-x
default:user::rwx
default:group::r-x
default:other::---
```

Die Vorgabe-ACL hat keinen Einfluss auf den Zugriff auf dieses Verzeichnis! Zukünftig angelegte Elemente aber erhalten (Abbildung 3)

a) eine Zugriffs-ACL nach dieser Vorgabe (bei Dateien werden die x-Bits allerdings nicht gesetzt)

b) falls es sich um Verzeichnisse handelt: eine Kopie dieser Vorgabe-ACL

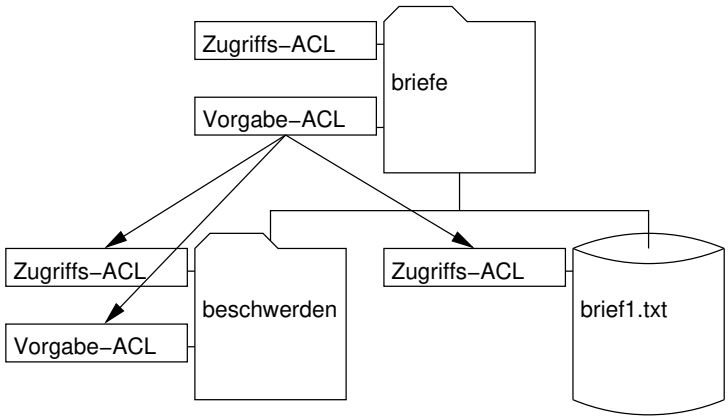

Abbildung 3: Vorgabe-ACLs

Den ersten Fall kann man hier sehen:

Terminal
```
schueler@debian964:~$ cal > briefe/brief1.txt
schueler@debian964:~$ getfacl briefe/brief1.txt
# file: briefe/brief1.txt
# owner: schueler
# group: schule
user::rw-
group::r--
other::---
```

Den zweiten Fall (Vererbung der Vorgabe-ACL) sieht man hier:

Terminal
```
schueler@debian964:~$ mkdir briefe/beschwerden
schueler@debian964:~$ getfacl briefe/beschwerden/
# file: briefe/beschwerden
# owner: schueler
# group: schule
user::rwx
group::r-x
other::---
default:user::rwx
default:group::r-x
default:other::---
```

Mit der Option `--default` des `setfacl`-Befehls kann man Vorgabe-ACLs auch ohne großen Aufwand erstellen. So erzeugt die Zeile

Terminal
```
schueler@debian964:~$ mkdir blabla;  setfacl --default --modify \
   other::- blabla
```

eine Vorgabe-ACL für dieses Verzeichnis aus:

- entweder der bisherigen Vorgabe-ACL (falls sie schon vorhanden war),
- oder (andernfalls) aus der bisherigen Zugriffs-ACL,
- modifiziert mit dem Befehl `modify other::-`.