Benutzerverwaltung unter Linux

Autor: Heiko Degenhardt (*hede@pingos.org*)
Layout: Matthias Hagedorn (*matthias.hagedorn@selflinux.org*)

Dieses Dokument beschreibt das Benutzerkonzept unter Linux. Es ist als Einführung gedacht. Die Benutzerverwaltung sowie alle daraus resultierenden Aufgaben werden in späteren Kapiteln behandelt.

Inhaltsverzeichnis

1 Einleitung

2 Grundlagen

3 Vorteile von Multi-User-Systemen

4 Das Linux-Benutzerkonzept

5 Wichtige Befehle

5.1 ls -al - Anzeigen von Datei- und Verzeichniseigenschaften
5.2 chmod - Ändern der Dateizugriffsrechte
5.3 chown - Ändern von Eigentümer und Gruppe von Dateien und Verzeichnissen
5.4 chgrp - Ändern der Gruppenzugehörigkeit von Dateien oder Verzeichnissen
5.5 id - Herausfinden der effektiven UIDs und GIDs
5.6 groups - Herausfinden, zu welchen Gruppen man gehört
5.7 Herausfinden, wer am System eingeloggt ist
5.8 Herausfinden von Informationen über laufende Prozesse, offene Dateien etc.
5.9 Benutzerverwaltung

6 Hinweise

1 Einleitung

Dieses Dokument beschreibt das Benutzerkonzept unter Linux. Es ist als Einführung gedacht. Die Benutzerverwaltung sowie alle daraus resultierenden Aufgaben werden in späteren Kapiteln behandelt.

2 Grundlagen

Zu den wichtigsten Eigenschaften von Linux zählt sicher die Fähigkeit zu echtem Multi-Tasking und zum Multi-User-Betrieb. **Multi-Tasking** bedeutet dabei, dass mehrere Prozesse (oder auch **Tasks**) parallel ablaufen, und sich somit die Ressourcen teilen. Mit **Multi-User** ist gemeint, dass mehrere Benutzer gleichzeitig am System arbeiten können, somit also das Multi-Tasking ausnutzen. Hierbei werden die einzelnen Benutzer, die Prozesse/Tasks, die sie starten und auch die von ihnen angelegten Dateien/Verzeichnisse vom Betriebssystem getrennt und die Zugriffe darauf kontrolliert.

In der Windows-Welt ist dieser Multi-User-Betrieb eigentlich kaum bekannt. Hier kann man zwar z.B. auch mehrere Benutzer auf dem System anlegen. Diese können aber (im Normalfall) nicht **parallel** am System arbeiten.

3 Vorteile von Multi-User-Systemen

Natürlich erscheint ein Multi-User-System nicht auf Anhieb von Vorteil auch für den Privatanwender zu sein. In erster Linie denkt man hierbei sicher an Server, auf die viele Kunden gleichzeitig Zugriff über Shell-Accounts, FTP, WWW oder was auch immer haben. Trotzdem bringt dieses Konzept auch für **normale** Anwender entscheidende Vorteile:

* Ein solches System führt zu einer disziplinierteren Arbeit (man kann sehr genau darauf achten, unter welchem Account man welche Aufgaben erledigt).
* Die Sicherheit ist erheblich erhöht, da die Zugriffsrechte auf Daten, Prozesse etc. je nach Benutzer und Nutzergruppe kontrolliert und eingeschränkt werden können.
  (Dies macht es z.B. möglich, dass selbst wenn sich ein Benutzer etwa einen Virus **einfangen** würde, dieser nur Zugriff auf die Daten des Benutzers hätte. Natürlich ist dafür das o.g. disziplinierte Arbeiten Voraussetzung)

4 Das Linux-Benutzerkonzept

Unter Linux werden alle Benutzer, die am System arbeiten können sollen, zu Benutzergruppen zusammengefasst. Alle Ressourcen können nun daraufhin freigegeben werden, welche Gruppe oder welcher Benutzer mit ihnen arbeiten können soll. Diese Rechtevergabe unterscheidet dabei nach:

* Der sog. User-ID (uid)
  Dies ist die ID, mit der sich ein Benutzer am System anmeldet.
* Der effektiven User-ID (euid)
  Dies ist die ID, mit der versucht wird, auf eine Resource zuzugreifen.
* Der Group-ID (gid)
  Die Entsprechung der uid für die Gruppenzugehörigkeit.
* Der effektiven Group-ID (egid)
  Dies entspricht der euid auf Gruppenbasis.

Für die Zugriffskontrolle überprüft das System bei jedem Zugriff auf eine Resource, unter welcher effektiven uid/guid der Zugriff erfolgt. Somit ist es z.B. möglich, unter der uid eines Benutzers zu arbeiten, aber trotzdem (temporär) auf Dateien zuzugreifen, die anderen Gruppen gehören.

Beispiel:

Beim Ändern des Passworts eines Benutzers muss das Programm `passwd` z.B. auf die Datei `/etc/shadow` zugreifen, auf die normalerweise kein Benutzer die notwendigen Rechte hat. Trotzdem kann das Programm `passwd` diese Dateien lesen, da es **zwischenzeitlich** einmal von der `uid` des Benutzers (der es aufruft) zur `uid` von root wechselt, und damit über die notwendigen Rechte verfügt. Dies ist möglich, da die auführbare Datei `/usr/bin/passwd` das sog. **set uid**-Bit gesetzt hat:

```
user@linux ~/temp/ # ls -al /usr/bin/passwd
-rwsr-xr-x    1 root     root        24680 Apr  7 17:59
```

(dies wird durch das erste "s" angezeigt).

Für weitere Informationen hierzu sei auf `info chmod` und `man chmod` verwiesen.

5 Wichtige Befehle

5.1 ls -al - Anzeigen von Datei- und Verzeichniseigenschaften

Beispiel:

```
user@linux ~/temp/ # ls -al

total 8
drwxr-xr-x    2 hede     hede         4096 Jul 10 07:25 .
drwxr-xr-x    8 hede     hede         4096 Jul 10 07:25 ..
-rw-r--r--    1 hede     hede            0 Jul 10 07:25 test.txt
```

Die Datei test.txt gehört dem Benutzer hede und der Gruppe hede. Der Eigentümer darf lesen und schreiben, die Gruppe und alle anderen nur lesen.

5.2 chmod - Ändern der Dateizugriffsrechte

Beispiel:

> Um dafür zu sorgen, dass alle Mitglieder der Gruppe auch schreibend auf die Datei zugreifen dürfen, und kein anderer lesen darf, gibt man ein:

```
user@linux ~/temp/ # chmod 660 test.txt
```

Überprüfung:

```
user@linux ~/temp/ # ls -al

total 8
drwxr-xr-x     2 hede      hede          4096 Jul 10 07:25 .
drwxr-xr-x     8 hede      hede          4096 Jul 10 07:25 ..
-rw-rw----     1 hede      hede             0 Jul 10 07:25 test.txt
```

5.3 chown - Ändern von Eigentümer und Gruppe von Dateien und Verzeichnissen

Soll nun z.B. die Gruppe floppy der Besitzer der Datei werden, gibt man ein:

```
user@linux ~/temp/ # chown hede:floppy test.txt
```

Test:

```
user@linux ~/temp/ # ls -al

total 8
drwxr-xr-x     2 hede      hede          4096 Jul 10 07:25 .
drwxr-xr-x     8 hede      hede          4096 Jul 10 07:25 ..
-rw-rw----     1 hede      floppy           0 Jul 10 07:25 test.txt
```

Um den Eigentümer und die Gruppe zu ändern, braucht man ausreichende Rechte. Meist muss dies also der Root-Benutzer tun. Prinzipiell muss man der Gruppe angehören, die man zum Eigentümer der Datei machen möchte.

5.4 chgrp - Ändern der Gruppenzugehörigkeit von Dateien oder Verzeichnissen

Auch mit diesem Befehl kann man die Gruppenzugehörigkeit von Dateien etc. ändern. Soll nun doch die Gruppe users der Eigentümer unserer Test-Datei werden, erreicht man dies durch:

```
user@linux ~/temp/ # chgrp users test.txt
```

Nachweis:

```
user@linux ~/temp/ # ls -al

total 8
drwxr-xr-x     2 hede      hede          4096 Jul 10 07:25 .
drwxr-xr-x     8 hede      hede          4096 Jul 10 07:25 ..
-rw-rw----     1 hede      users            0 Jul 10 07:25 test.txt
```

Auch hierfür braucht man die entsprechenden Rechte.

5.5 id - Herausfinden der effektiven UIDs und GIDs

Möchte man Informationen darüber erhalten, unter welcher UID man eingeloggt ist, zu welchen Gruppen man gehört, was die primäre Gruppe ist etc., kann man den Befehl `id` verwenden:

```
user@linux ~/temp/ # id

uid=1000(hede) gid=1000(hede)
groups=1000(hede),25(floppy),100(users)
```

5.6 groups - Herausfinden, zu welchen Gruppen man gehört

Mit dem Befehl "groups" kann man sich anzeigen lassen, zu welchen Gruppen man gehört:

```
user@linux ~/temp/ # groups

hede floppy users
```

Diese Informationen kann man sich auch für andere Benutzer anzeigen lassen:

```
user@linux ~/temp/ # groups root

root : root
```

5.7 Herausfinden, wer am System eingeloggt ist

Um mal zu sehen, wer gerade am System arbeitet, kann man folgendes Kommando verwenden:

```
user@linux ~/temp/ # w

07:37:58 up 15 days, 49 min,  5 users, load average: 0.05, 0.08 0.03
USER    TTY       FROM       LOGIN@    IDLE    JCPU    PCPU    WHAT
root    tty1       -         25Jun02   6:20    0.82s   0.82s   -bash
hede    tty2       -         02Jul02   7days   0.30s   0.30s   -bash
hede    tty3       -         Thu16     5days   0.21s   0.21s   -bash
hede    pts/0    www2        Mon11     3.00s   3.94s   3.45s   vim t.txt
hede    pts/1    www2        07:32     0.00s   0.13s   0.06s   w
```

Damit kann man also auch sehen, woher jemand eingeloggt ist, und was er gerade macht.

Eine verkürztere Information liefert der Befehl who:

```
user@linux ~/temp/ # who

root     tty1         Jun 25 06:49
hede     tty2         Jul  2 07:47
hede     tty3         Jul  4 16:28
hede     pts/0        Jul  8 11:45 (www2.sentec-elektronik.de)
hede     pts/1        Jul 10 07:32 (www2.sentec-elektronik.de)
```

Ist man sich nun nicht mehr sicher, wer man selbst ist, hilft einem Linux auch dabei.
Man gibt einfach `whoami` ein und bekommt die aktuelle uid angezeigt:

```
user@linux ~/temp/ # hede@www:~/temp$ whoami

hede
```

5.8 Herausfinden von Informationen über laufende Prozesse, offene Dateien etc.

> HINWEIS: Ich werde hier zu diesen Befehl keine Beispiele mehr einfügen, da dies einerseits das Dokument zu sehr aufblähen würde, und andererseits diese Befehle leicht ausprobiert werden können!

Mit `lsof` (etwa **list open files**) kann man ermitteln, welche Dateien gerade geöffnet sind, und von wem.

`ps aux` zeigt laufenden Prozesse (mit zusätzlichen Infos über Eigentümer, Ressourcenverbrauch etc.) an.

5.9 Benutzerverwaltung

Für die Benutzerverwaltung, also das Anlegen, Ändern, Löschen von Benutzern und Gruppen, dienen die folgenden Befehle:

`useradd`	Neuen Benutzer anlegen
`usermod`	Existierenden Benutzer verändern
`groupadd`	Gruppe anlegen
`groupmod`	Existierende Gruppe verändern

Diese werden in einem eigenen Kapitel von *SelfLinux* behandelt, weshalb ich hier nicht näher darauf eingehen möchte.

Ausserdem gibt es (je nach Distribution) noch weitere Möglichkeiten, diese Verwaltungsaufgaben zu erledigen. So bietet bspw. *Debian* die Tools `adduser`, `addgroup`, `deluser`, `delgroup` an. Unter *SuSE Linux* kann man diese Aufgaben auch bequem mit `yast` erledigen.

6 Hinweise

Hier folgen noch ein paar (wenige) Hinweise für ein **vernünftiges** Arbeiten unter Ausnutzung des Benutzerkonzepts:

* Man sollte grundsätzlich nur dann als Root arbeiten, wenn es unbedingt notwendig ist!
  Dies hat den Vorteil, dass man nicht **aus Versehen** mal zu viele wichtige Dateien löscht. Ausserdem muss man sich vorstellen, dass alle Prozesse (also bspw. auch Mail-Tools), die unter dem Root-Account laufen, _vollen_ Zugriff auf das gesamte System haben! Tritt also dabei ein Fehler auf, oder enthält das Programm z.B. Code, wie er z.B. auf anderen Systemen als Virus oder Trojaner vorkommt, ist das gesamte System gefährdet!
* Alle Dienste/Server, die man betreibt, sollten mit so wenig Rechten wie möglich laufen!
  Dies ist bei den meisten Systemen standardmässig so eingestellt.
* Als Benutzer sollte man sich gut überlegen, welche Zugriffsrechte man seinen Dateien oder Verzeichnissen gibt!
  Dies hat nicht nur die Schutzfunktion für die Privatssphäre, sondern auch sichheitstechnische Aspekte!