# 4.1 Benutzer, Gruppen und Rechte

Linux ist ein Mehrbenutzersystem.

Das bedeutet:

```
Mehrere Benutzer können auf demselben System arbeiten,
ohne automatisch Zugriff auf alle Dateien,
Prozesse
oder Systemfunktionen zu haben.
```

Dafür nutzt Linux:

```
- Benutzer
- Gruppen
- Besitzer
- Dateirechte
- sudo
- Root-Rechte
- optional ACLs
```

Für Fachinformatiker Systemintegration ist dieses Thema besonders wichtig, weil Rechteprobleme, Sicherheitskonzepte, Dienstkonten, SSH-Zugriffe und Datei-Freigaben sehr häufig mit Benutzer- und Gruppenrechten zusammenhängen.

Merksatz:

```
Linux-Sicherheit beginnt bei Benutzern,
Gruppen
und Rechten.
```

---

**Lernziele**

Nach dieser Seite solltest du erklären können:

```
- was Benutzer unter Linux sind
- was Gruppen sind
- was UID und GID bedeuten
```

- wie Linux Dateirechte aufteilt
- was Besitzer,
  Gruppe
  und Andere bedeuten
- was r,
  w
  und x bedeuten
- wie chmod,
  chown
  und chgrp grundsätzlich funktionieren
- was sudo macht
- warum Root-Rechte gefährlich sein können
- welche typischen Rechtefehler auftreten

---

### Grundidee

Jede Datei und jedes Verzeichnis unter Linux hat Berechtigungen.

Diese Berechtigungen regeln:

- wer lesen darf
- wer schreiben darf
- wer ausführen darf
- wem die Datei gehört
- welcher Gruppe die Datei zugeordnet ist

Linux unterscheidet dabei klassisch drei Bereiche:

| Bereich | Bedeutung |
|---|---|
| Benutzer / Besitzer | Eigentümer der Datei |
| Gruppe | zugeordnete Benutzergruppe |
| Andere | alle übrigen Benutzer |

Merksatz:

Linux-Rechte gelten für Besitzer,
Gruppe
und Andere.

---

**Benutzer**

Ein Benutzer ist ein Konto auf dem Linux-System.

Benutzer können:

```
- sich anmelden
- Dateien besitzen
- Prozesse starten
- Gruppen zugeordnet sein
- Rechte erhalten
- Dienste ausführen
```

Beispiele:

```
felix

root

www-data

postgres

backup
```

Wichtig:

```
Nicht jeder Benutzer ist ein Mensch.
Viele Dienste laufen unter eigenen Systembenutzern.
```

Merksatz:

```
Benutzer können Menschen oder Dienstkonten sein.
```

---

**Root**

root ist der administrative Hauptbenutzer unter Linux.

root darf fast alles:

```
- Systemdateien ändern
- Benutzer anlegen
- Rechte ändern
- Dienste starten und stoppen
- Software installieren
- Dateien anderer Benutzer lesen oder löschen
- Systemkonfiguration ändern
```

Das ist mächtig, aber auch gefährlich.

Ein falscher Befehl mit Root-Rechten kann das System beschädigen.

Merksatz:

```
Root-Rechte nur bewusst verwenden.
```

---

**Normale Benutzer**

Normale Benutzer haben eingeschränkte Rechte.

Sie dürfen typischerweise:

```
- im eigenen Home-Verzeichnis arbeiten
- eigene Dateien bearbeiten
- eigene Prozesse starten
- Programme ausführen
```

Sie dürfen normalerweise nicht:

```
- Systemdateien ändern
- Dienste systemweit verwalten
- fremde Dateien beliebig lesen
- Pakete installieren
- Benutzer anlegen
```

Vorteil:

```
Fehler oder Angriffe wirken sich weniger stark auf das gesamte System aus.
```

Merksatz:

> Normale Benutzer begrenzen Risiken.

---

### Systembenutzer und Dienstkonten

Viele Dienste laufen unter eigenen Benutzern.

Beispiele:

| Benutzer | Typischer Zweck |
|---|---|
| www-data | Webserver-Prozesse |
| nginx | Webserver-Prozesse |
| postgres | PostgreSQL-Datenbank |
| mysql | MySQL/MariaDB-Datenbank |
| backup | Backup-Aufgaben |
| nobody | stark eingeschränkter Benutzer |

Warum?

> Dienste sollen nicht mit Root-Rechten laufen,
> wenn das nicht notwendig ist.

Vorteil:

> Wenn ein Dienst kompromittiert wird,
> ist der Schaden begrenzter.

Merksatz:

> Dienste sollten nur die Rechte haben,
> die sie benötigen.

---

### UID und GID

Linux identifiziert Benutzer und Gruppen intern über Zahlen.

UID bedeutet:

User ID

GID bedeutet:

Group ID

Beispiele:

| Begriff | Bedeutung |
|---|---|
| UID | eindeutige Benutzerkennung |
| GID | eindeutige Gruppenkennung |
| UID 0 | root-Benutzer |
| normale UID | normaler Benutzer oder Systembenutzer |

Wichtig:

Der Benutzername ist für Menschen lesbar.
Intern arbeitet Linux stark mit UID und GID.

Merksatz:

UID identifiziert Benutzer.
GID identifiziert Gruppen.

---

### Benutzerinformationen anzeigen

Aktuellen Benutzer anzeigen:

```
whoami
```

Benutzer-ID und Gruppen anzeigen:

```
id
```

Beispiel:

```
id felix
```

Mögliche Ausgabe:

```
uid=1000(felix) gid=1000(felix) groups=1000(felix),27(sudo)
```

Bedeutung:

```
Benutzer felix hat UID 1000,
Hauptgruppe felix
und ist zusätzlich in der Gruppe sudo.
```

Merksatz:

```
id zeigt Benutzer,
UID,
GID
und Gruppen.
```

---

**Gruppen**

Gruppen fassen Benutzer zusammen.

Damit können Rechte einfacher vergeben werden.

Beispiel:

```
Alle Benutzer der Gruppe developers
dürfen auf ein Projektverzeichnis zugreifen.
```

Vorteile:

```
Rechte müssen nicht einzeln pro Benutzer vergeben werden

Verwaltung wird übersichtlicher

Zugriff kann über Gruppenmitgliedschaft gesteuert werden
```

```
Rollen lassen sich besser abbilden
```

Merksatz:

```
Gruppen vereinfachen Rechteverwaltung.
```

---

### Hauptgruppe und Zusatzgruppen

Ein Benutzer hat eine Hauptgruppe und kann zusätzlich Mitglied weiterer Gruppen sein.

Beispiel:

```
Benutzer:
felix

Hauptgruppe:
felix

Zusatzgruppen:
sudo,
docker,
www-data
```

Die Hauptgruppe wird oft für neu erstellte Dateien verwendet.

Zusatzgruppen geben weitere Rechte.

Merksatz:

```
Hauptgruppe ist Standardgruppe.
Zusatzgruppen geben zusätzliche Rechte.
```

---

### Wichtige Dateien für Benutzer und Gruppen

Linux speichert Benutzer- und Gruppeninformationen in bestimmten Dateien.

| Datei | Bedeutung |
| --- | --- |
| /etc/passwd | Benutzerkonten |

| Datei | Bedeutung |
|---|---|
| /etc/group | Gruppen |
| /etc/shadow | Passwort-Hashes und Passwortinformationen |
| /etc/sudoers | sudo-Regeln |
| /etc/sudoers.d/ | zusätzliche sudo-Regeln |

Wichtig:

```
/etc/shadow ist besonders geschützt,
weil dort Passwort-Hashes liegen.
```

Merksatz:

```
Benutzer stehen in /etc/passwd,
Gruppen in /etc/group.
```

---

### /etc/passwd

Die Datei /etc/passwd enthält Benutzerinformationen.

Beispielzeile:

```
felix:x:1000:1000:Felix Ulrich:/home/felix:/bin/bash
```

Bedeutung:

| Feld | Bedeutung |
|---|---|
| felix | Benutzername |
| x | Passwort liegt nicht direkt hier |
| 1000 | UID |
| 1000 | GID |
| Felix Ulrich | Kommentar oder Beschreibung |
| /home/felix | Home-Verzeichnis |
| /bin/bash | Login-Shell |

Merksatz:

```
/etc/passwd enthält Benutzerstammdaten.
```

---

### /etc/group

Die Datei /etc/group enthält Gruppeninformationen.

Beispielzeile:

```
sudo:x:27:felix
```

Bedeutung:

| Feld | Bedeutung |
|---|---|
| sudo | Gruppenname |
| x | Passwortfeld, meist ungenutzt |
| 27 | GID |
| felix | Mitglieder der Gruppe |

Merksatz:

```
/etc/group zeigt Gruppen und Mitglieder.
```

---

### Dateirechte anzeigen

Dateirechte zeigt man mit:

```
ls -l
```

Beispiel:

```
-rw-r--r-- 1 felix users 1200 Jul 08 12:30 notiz.txt
```

Wichtiger Teil:

```
-rw-r--r--
```

Aufteilung:

```
-    rw-    r--    r--
Typ  User   Gruppe Andere
```

Merksatz:

```
ls -l zeigt Dateityp,
Rechte,
Besitzer
und Gruppe.
```

---

### Rechtebereiche

Linux-Rechte sind klassisch in drei Bereiche aufgeteilt:

| Bereich | Bedeutung |
|---|---|
| User / Owner | Besitzer der Datei |
| Group | zugeordnete Gruppe |
| Others | alle anderen Benutzer |

Beispiel:

```
-rw-r--r--
```

Bedeutung:

| Bereich | Rechte |
|---|---|
| Besitzer | rw- |
| Gruppe | r-- |
| Andere | r-- |

Merksatz:

```
Rechte werden in Dreiergruppen gelesen.
```

---

### r, w und x

Die Buchstaben bedeuten:

| Zeichen | Name | Bei Dateien | Bei Verzeichnissen |
|---|---|---|---|
| r | read | Datei lesen | Inhalt auflisten |
| w | write | Datei ändern | Einträge erstellen/löschen/umbenennen |
| x | execute | Datei ausführen | Verzeichnis betreten |

Wichtig:

```
x hat bei Dateien und Verzeichnissen unterschiedliche praktische Bedeutung.
```

Merksatz:

```
r = lesen.
w = schreiben.
x = ausführen oder betreten.
```

---

### Rechte bei Dateien

Bei Dateien bedeutet:

| Recht | Bedeutung |
|---|---|
| r | Inhalt lesen |
| w | Inhalt ändern |
| x | Datei ausführen |

Beispiel:

```
-rwxr-xr--
```

Bedeutung:

```
Besitzer darf lesen,
schreiben
und ausführen.

Gruppe darf lesen
und ausführen.
```

Andere dürfen nur lesen.

Merksatz:

x macht eine Datei ausführbar.

---

### Rechte bei Verzeichnissen

Bei Verzeichnissen bedeutet:

| Recht | Bedeutung |
|---|---|
| r | Inhalt des Verzeichnisses anzeigen |
| w | Dateien im Verzeichnis erstellen, löschen oder umbenennen |
| x | Verzeichnis betreten und Pfade darin nutzen |

Wichtig:

Ohne x kann man ein Verzeichnis nicht sinnvoll betreten,
auch wenn r gesetzt ist.

Beispiel:

drwxr-x---

Bedeutung:

Besitzer darf alles.

Gruppe darf betreten und auflisten.

Andere haben keinen Zugriff.

Merksatz:

Bei Verzeichnissen ist x besonders wichtig.

---

### Numerische Rechte

Rechte können auch als Zahlen dargestellt werden.

| Recht | Wert |
|---|---|
| r | 4 |
| w | 2 |
| x | 1 |

Kombinationen:

| Rechte | Zahl |
|---|---|
| --- | 0 |
| --x | 1 |
| -w- | 2 |
| -wx | 3 |
| r-- | 4 |
| r-x | 5 |
| rw- | 6 |
| rwx | 7 |

Beispiele:

| Zahl | Bedeutung |
|---|---|
| 644 | Besitzer rw-, Gruppe r--, Andere r-- |
| 600 | Besitzer rw-, Gruppe ---, Andere --- |
| 755 | Besitzer rwx, Gruppe r-x, Andere r-x |
| 700 | Besitzer rwx, Gruppe ---, Andere --- |

Merksatz:

```
r=4,
w=2,
x=1.
```

---

### chmod

Mit chmod ändert man Dateirechte.

Beispiele:

```
chmod 644 datei.txt

chmod 600 geheim.txt

chmod 755 script.sh

chmod +x script.sh
```

Bedeutung:

| Befehl | Bedeutung |
|---|---|
| chmod 644 datei.txt | typische Rechte für normale Textdatei |
| chmod 600 geheim.txt | nur Besitzer darf lesen und schreiben |
| chmod 755 script.sh | Besitzer darf alles, andere lesen/ausführen |
| chmod +x script.sh | Ausführrecht hinzufügen |

Merksatz:

```
chmod ändert Rechte.
```

---

**chown**

Mit chown ändert man den Besitzer einer Datei.

Beispiel:

```
chown felix datei.txt
```

Besitzer und Gruppe ändern:

```
chown felix:users datei.txt
```

Rekursiv für Verzeichnisse:

```
chown -R felix:users projektordner
```

Wichtig:

```
chown benötigt häufig administrative Rechte.
```

Merksatz:

```
chown ändert Besitzer und optional Gruppe.
```

---

**chgrp**

Mit chgrp ändert man die Gruppe einer Datei.

Beispiel:

```
chgrp developers projekt.txt
```

Rekursiv:

```
chgrp -R developers projektordner
```

Merksatz:

```
chgrp ändert die Gruppenzuordnung.
```

---

**sudo**

sudo erlaubt berechtigten Benutzern, einzelne Befehle mit erhöhten Rechten auszuführen.

Beispiel:

```
sudo systemctl restart ssh
```

Vorteile:

```
keine dauerhafte Root-Sitzung nötig

bessere Nachvollziehbarkeit
```

gezielte administrative Rechte

geringeres Risiko als dauerhaft als root zu arbeiten

Wichtig:

sudo sollte nur berechtigten Benutzern gegeben werden.

Merksatz:

sudo gibt gezielt erhöhte Rechte für einzelne Befehle.

---

**su**

su steht für:

substitute user

Mit su kann man zu einem anderen Benutzer wechseln.

Beispiel:

su -

Häufig wird damit zu root gewechselt, wenn das Root-Konto aktiviert ist.

Unterschied zu sudo:

| Befehl | Bedeutung |
|---|---|
| sudo befehl | einzelnen Befehl mit erhöhten Rechten ausführen |
| su | Benutzer wechseln |
| su - | Login-Shell des Zielbenutzers starten |

Merksatz:

sudo führt Befehl aus.
su wechselt Benutzer.

### Dateien ausführbar machen

Ein Skript muss ausführbar sein, wenn man es direkt starten möchte.

Beispiel:

```
chmod +x backup.sh
```

Starten:

```
./backup.sh
```

Wichtig:

```
./ bedeutet:
Datei im aktuellen Verzeichnis ausführen.
```

Ohne Ausführrecht kommt häufig:

```
Permission denied
```

Merksatz:

```
Skript direkt starten braucht x-Recht.
```

---

### Typische Rechte für Dateien

| Zweck | Rechte | Bedeutung |
|---|---|---|
| normale Textdatei | 644 | Besitzer schreibt, andere lesen |
| private Datei | 600 | nur Besitzer liest und schreibt |
| Skript oder Programm | 755 | Besitzer schreibt, alle dürfen ausführen |
| geheimes Skript | 700 | nur Besitzer darf alles |
| öffentlich lesbares Verzeichnis | 755 | andere können betreten und lesen |
| privates Verzeichnis | 700 | nur Besitzer Zugriff |

Merksatz:

644 für normale Dateien.
755 für ausführbare Programme oder Verzeichnisse.
600 für private Dateien.

---

### Rechte und Sicherheit

Zu breite Rechte sind ein Sicherheitsrisiko.

Beispiel kritisch:

```
chmod 777 datei
```

777 bedeutet:

```
Besitzer darf alles.

Gruppe darf alles.

Andere dürfen alles.
```

Risiko:

```
Jeder Benutzer kann lesen,
schreiben
und ausführen.
```

Besser:

```
Rechte gezielt setzen.
```

Merksatz:

```
777 ist selten eine saubere Lösung.
```

---

### Permission denied

Die Meldung:

```
Permission denied
```

bedeutet:

```
Zugriff verweigert.
```

Mögliche Ursachen:

```
Datei nicht lesbar

Datei nicht beschreibbar

Datei nicht ausführbar

Verzeichnis nicht betretbar

falscher Besitzer

falsche Gruppe

sudo nötig
```

Prüfen:

```
ls -l datei

ls -ld verzeichnis

id
```

Merksatz:

```
Permission denied zuerst mit ls -l und id prüfen.
```

---

**Rechteproblem bei Verzeichnissen**

Ein Benutzer hat Leserecht auf eine Datei, kann sie aber trotzdem nicht öffnen.

Mögliche Ursache:

```
fehlendes x-Recht auf einem übergeordneten Verzeichnis.
```

Beispiel:

```
/srv/projekt/datei.txt
```

Damit der Benutzer die Datei erreichen kann, braucht er passende Rechte auf:

```
/srv

/srv/projekt

/srv/projekt/datei.txt
```

Merksatz:

```
Für Zugriff auf Dateien müssen auch Verzeichnisrechte passen.
```

---

**Gruppenbasierter Zugriff**

Beispiel:

```
Mehrere Benutzer sollen auf /srv/projekt zugreifen.
```

Sinnvolles Vorgehen:

```
Gruppe projekt anlegen

Benutzer zur Gruppe hinzufügen

Verzeichnis der Gruppe projekt zuordnen

Gruppenrechte passend setzen
```

Beispielhafte Befehle:

```
chgrp -R projekt /srv/projekt

chmod -R 770 /srv/projekt
```

Bedeutung:

```
Besitzer und Gruppe dürfen alles.
Andere haben keinen Zugriff.
```

Merksatz:

```
Gemeinsamer Zugriff wird sauber über Gruppen geregelt.
```

---

**Benutzer anlegen**

Typische Befehle je nach Distribution:

```
useradd

adduser
```

Beispiel:

```
adduser felix
```

oder:

```
useradd -m felix
```

Bedeutung:

```
-m erstellt ein Home-Verzeichnis.
```

Wichtig:

```
Die genaue Bedienung kann je nach Distribution unterschiedlich sein.
```

Merksatz:

```
Benutzerverwaltung kann je nach Distribution leicht unterschiedlich sein.
```

---

### Passwort setzen

Passwort für Benutzer setzen:

```
passwd felix
```

Eigenes Passwort ändern:

```
passwd
```

Wichtig:

```
Starke Passwörter verwenden.

Für administrative Zugänge zusätzlich MFA oder Schlüsselverfahren nutzen,
wenn möglich.
```

Merksatz:

```
passwd setzt oder ändert Passwörter.
```

---

### Benutzer zu Gruppe hinzufügen

Beispiel:

```
usermod -aG gruppe benutzer
```

Wichtig:

```
-aG bedeutet:
zur Zusatzgruppe hinzufügen,
ohne bestehende Gruppen zu entfernen.
```

Beispiel:

```
usermod -aG sudo felix
```

Nach Gruppenänderungen ist oft eine neue Anmeldung erforderlich.

Merksatz:

```
usermod -aG fügt Benutzer zu Zusatzgruppe hinzu.
```

---

**Benutzer löschen**

Benutzer löschen:

```
userdel benutzer
```

Benutzer inklusive Home-Verzeichnis löschen:

```
userdel -r benutzer
```

Wichtig:

```
Vor dem Löschen prüfen:

Gibt es noch Dateien des Benutzers?

Gibt es laufende Prozesse?

Gibt es Dienste oder Cronjobs?

Müssen Daten archiviert werden?
```

Merksatz:

```
Benutzer nicht löschen,
ohne Daten und Dienste zu prüfen.
```

---

### ACLs

ACL steht für:

```
Access Control List
```

ACLs erlauben feinere Rechte als klassische Linux-Rechte.

Beispiel:

```
Eine bestimmte Benutzerin soll Zugriff erhalten,
ohne Besitzer oder Gruppe zu ändern.
```

Typische Befehle:

```
getfacl datei

setfacl -m u:benutzer:r datei
```

Wichtig:

```
ACLs sind nützlich,
können Rechteverwaltung aber komplexer machen.
```

Merksatz:

```
ACLs erweitern klassische Linux-Rechte.
```

---

### SetUID, SetGID und Sticky Bit

Neben normalen Rechten gibt es Sonderrechte.

| Sonderrecht | Bedeutung |
|---|---|
| SetUID | Programm läuft mit Rechten des Dateibesitzers |
| SetGID | Programm oder Verzeichnis nutzt Gruppenlogik besonders |
| Sticky Bit | in Verzeichnissen dürfen Benutzer nur eigene Dateien löschen |

Typisches Beispiel für Sticky Bit:

```
/tmp
```

Rechte häufig:

```
drwxrwxrwt
```

Das t am Ende zeigt das Sticky Bit.

Merksatz:

```
Sticky Bit schützt gemeinsame Verzeichnisse wie /tmp.
```

---

### Typische Fehlerbilder

| Fehlerbild | Wahrscheinliche Ursache |
|---|---|
| Permission denied beim Lesen | fehlendes Leserecht |
| Permission denied beim Schreiben | fehlendes Schreibrecht |
| Skript startet nicht | fehlendes Ausführrecht |
| Datei trotz Rechten nicht erreichbar | Verzeichnisrecht fehlt |
| Benutzer sieht Datei nicht | fehlendes r auf Verzeichnis |
| Benutzer kann Datei nicht löschen | fehlendes w auf Verzeichnis |
| sudo nicht erlaubt | Benutzer nicht in sudoers |
| Gruppenrecht wirkt nicht | Benutzer neu anmelden oder falsche Gruppe |
| Dienst kann Datei nicht lesen | Dienstbenutzer hat keine Rechte |
| 777 wurde gesetzt | unsichere Notlösung |

---

### Sichere Arbeitsweise bei Rechteproblemen

Nicht sofort:

```
chmod 777
```

Besser prüfen:

```
whoami

id

ls -l datei

ls -ld verzeichnis

groups benutzer

getfacl datei
```

Dann gezielt entscheiden:

```
Besitzer ändern?

Gruppe ändern?

Rechte ändern?

Benutzer in Gruppe aufnehmen?

sudo nötig?
```

Merksatz:

```
Rechteprobleme gezielt lösen,
nicht pauschal mit 777.
```

---

### Typische Prüfungsfragen

| Frage | Kurzantwort |
|---|---|
| Was ist root? | administrativer Hauptbenutzer |
| Was bedeutet UID? | User ID |
| Was bedeutet GID? | Group ID |
| Was macht chmod? | ändert Rechte |
| Was macht chown? | ändert Besitzer |

| Frage | Kurzantwort |
|---|---|
| Was macht chgrp? | ändert Gruppe |
| Was macht sudo? | führt Befehl mit erhöhten Rechten aus |
| Was bedeutet r? | lesen |
| Was bedeutet w? | schreiben |
| Was bedeutet x bei Dateien? | ausführen |
| Was bedeutet x bei Verzeichnissen? | betreten |
| Was bedeutet 755? | rwx r-x r-x |
| Was bedeutet 644? | rw- r-- r-- |
| Was bedeutet 600? | nur Besitzer lesen/schreiben |
| Warum ist 777 kritisch? | jeder darf alles |

---

## Typische Prüfungsfallen

| Falle | Richtig denken |
|---|---|
| x bei Verzeichnissen vergessen | x bedeutet Verzeichnis betreten |
| chmod 777 als Lösung | unsicher und meist falsch |
| nur Dateirechte prüfen | auch Verzeichnisrechte prüfen |
| sudo immer verwenden | erst Ursache prüfen |
| Besitzer und Gruppe verwechseln | beides getrennt betrachten |
| Gruppenänderung wirkt sofort erwarten | neue Anmeldung kann nötig sein |
| Dienst läuft als root | Sicherheitsrisiko, wenn nicht nötig |
| /etc/shadow frei lesbar erwarten | besonders geschützt |
| ACLs übersehen | zusätzliche Rechte können existieren |
| Root und normales Konto gleich behandeln | Root hat besondere Rechte |

---

## IHK-sichere Kurzformulierung

Linux ist ein Mehrbenutzersystem und steuert Zugriffe über Benutzer, Gruppen und Rechte. Jede Datei und jedes Verzeichnis hat einen Besitzer, eine zugeordnete Gruppe und Rechte für Besitzer, Gruppe und Andere. Die klassischen Rechte sind r für Lesen, w für Schreiben und x für Ausführen beziehungsweise bei Verzeichnissen für Betreten. Mit chmod werden Rechte geändert, mit chown der Besitzer und mit chgrp die Gruppe. sudo erlaubt berechtigten Benutzern, einzelne Befehle mit erhöhten Rechten auszuführen. Rechteprobleme sollten mit whoami, id, ls -l und ls -ld geprüft werden. Pauschale Rechte wie 777 sind unsicher und sollten vermieden werden.

---

## Merksätze

Linux ist ein Mehrbenutzersystem.

Benutzer können Menschen oder Dienste sein.

Root ist der administrative Hauptbenutzer.

UID identifiziert Benutzer.

GID identifiziert Gruppen.

Gruppen vereinfachen Rechteverwaltung.

Jede Datei hat Besitzer,
Gruppe
und Rechte.

Rechte gelten für Besitzer,
Gruppe
und Andere.

r bedeutet lesen.

w bedeutet schreiben.

x bedeutet bei Dateien ausführen.

x bedeutet bei Verzeichnissen betreten.

chmod ändert Rechte.

chown ändert Besitzer.

chgrp ändert Gruppe.

sudo gibt gezielt erhöhte Rechte.

su wechselt Benutzer.

644 ist typisch für normale Dateien.

600 ist typisch für private Dateien.

755 ist typisch für Programme oder Verzeichnisse.

700 ist typisch für private Verzeichnisse oder Skripte.

777 ist selten eine gute Lösung.

Permission denied bedeutet Rechteproblem.

Bei Dateien auch Verzeichnisrechte prüfen.

Dienstkonten sollten nur notwendige Rechte haben.

Gruppenänderungen können neue Anmeldung erfordern.

ACLs erweitern klassische Rechte.

Sticky Bit schützt gemeinsame Verzeichnisse wie /tmp.