

corewire

Linux

Benutzer, Gruppen und Rechte

Folien-Hinweis

- `Space`, `Page down`: Nächste Folie
- `Page up`: Vorherige Folie
- `ESC`, `o`: Übersicht

[Zur Kapitelübersicht](#)

Der root-Benutzer

root - Der Superuser

UID = 0: Höchste Berechtigung

- **Vollzugriff** auf alle Dateien und Verzeichnisse
- **Kein Berechtigungsscheck** durch das System
- **Systemverwaltung** und kritische Operationen
- **Mit großer Macht kommt große Verantwortung**

root - Der Superuser

```
# Root-Benutzer Information anzeigen
id root
# uid=0(root) gid=0(root) groups=0(root)

# Wechsel zu root (falls erlaubt)
sudo -i

# sudo führt Befehle mit root-Rechten aus
sudo ls -la /root
```

Warum ist UID=0 besonders?

Kernel-Level Privilegierung:

- **Kernel prüft nur die UID**, nicht den Benutzernamen
- **UID=0** bedeutet automatisch Superuser-Rechte
- **Umgehung aller Dateiberechtigungen**
- **Zugriff auf Kernel-Funktionen und Hardware**

Best Practices mit root

- **Prinzip der minimalen Rechte:** Nur wenn nötig
- **Separate Accounts** für tägliche Arbeit
- **sudo verwenden** statt direkter root-Anmeldung
- **Logging und Monitoring** von root-Aktivitäten

Benutzer- und Gruppenverwaltung

Grundlegende Befehle zur Identitätsfeststellung

```
# Aktueller Benutzername
```

```
whoami
```

```
# Detaillierte Benutzerinformationen
```

```
id
```

```
# uid=1000(user) gid=1000(user) groups=1000(user),4(adm),27(sudo)
```

```
# ID eines anderen Benutzers anzeigen
```

```
id username
```

Password-Management

```
# Eigenes Passwort ändern  
passwd
```

```
# Als Administrator: Benutzerpasswort ändern  
sudo passwd username
```

sudo - Rechte temporär erhöhen

```
# Einzelnen Befehl als root ausführen
sudo systemctl restart nginx

# Shell als root starten
sudo -i

# Als anderer Benutzer ausführen
sudo -u www-data touch /tmp/test

# Nutzer wechseln

sudo su - username
```

sudo-Konfiguration verstehen

```
# Mitgliedschaft in sudo-Gruppe prüfen  
groups  
id | grep sudo  
  
# User zur sudo-Gruppe hinzufügen  
sudo usermod -aG sudo username
```

Benutzer erstellen

adduser vs useradd

Feature	adduser	useradd
Typ	High-level Script	Low-level Befehl
Interaktiv	Ja	Nein
Home-Dir	Automatisch	Mit <code>-m</code> Flag
Skeleton-Dateien	Automatisch kopiert	Mit <code>-m</code> Flag
Passwort setzen	Während der Erstellung	Separater <code>passwd</code> Aufruf
Verfügbarkeit	Debian/Ubuntu	Linux

adduser - Der benutzerfreundliche Weg

Interactive Benutzererstellung:

```
# Neuen Benutzer interaktiv erstellen
sudo adduser newuser

# Beispiel-Session:
# Adding user 'newuser' ...
# Adding new group 'newuser' (1001) ...
# Adding new user 'newuser' (1001) with group 'newuser' ...
# Creating home directory '/home/newuser' ...
# Copying files from '/etc/skel' ...
# New password: ****
# Retype new password: ****
# Full Name []: Max Mustermann
# Room Number []:
# Work Phone []:
# Home Phone []:
# Other []:
# Is the information correct? [Y/n] Y
```

adduser - Weitere Optionen

Spezielle Benutzertypen mit adduser:

```
# System-Benutzer erstellen (für Services)
sudo adduser --system --no-create-home serviceuser

# Benutzer zu existierender Gruppe hinzufügen
sudo adduser --ingroup developers newdev

# Benutzer mit spezifischer Shell
sudo adduser --shell /bin/bash newuser

# Benutzer ohne Home-Verzeichnis
sudo adduser --no-create-home tempuser
```


useradd - Der direkte Weg

Nicht-interaktive Benutzererstellung:

```
# Minimaler Benutzer (nur Account)
sudo useradd newuser

# Benutzer mit Home-Verzeichnis und Standard-Shell
sudo useradd -m -s /bin/bash newuser

# Benutzer mit spezifischer UID und GID
sudo useradd -m -u 1500 -g 1500 newuser

# System-Benutzer für Services
sudo useradd -r -s /bin/false serviceuser
```

useradd - Detaillierte Optionen

Vollständige Benutzerkonfiguration:

```
# Benutzer mit allen Optionen
sudo useradd \
  -m \                # Home-Verzeichnis erstellen
  -d /home/custom \   # Spezifisches Home-Verzeichnis
  -s /bin/bash \       # Login-Shell
  -c "Full Name" \    # Kommentar/Vollständiger Name
  -u 1200 \            # Spezifische UID
  -g users \           # Primäre Gruppe
  -G sudo,docker \     # Zusätzliche Gruppen
  username

# Passwort separat setzen
sudo passwd username
```

Praktischer Vergleich

- **useradd**: Besser geeignet für Scripts und Automation
- **adduser**: Besser für manuelle, einmalige Benutzererstellung

Rechtekonzepte

Das Unix-Berechtigungsmodell

Drei Ebenen der Dateiberechtigung:

- **Owner (Besitzer):** Der Benutzer dem die Datei gehört
- **Group (Gruppe):** Mitglieder der Datei-Gruppe
- **Others (Andere):** Alle anderen Benutzer

Drei Arten von Berechtigungen:

- **Read (r):** Datei lesen / Verzeichnis auflisten
- **Write (w):** Datei schreiben / Verzeichnis ändern
- **Execute (x):** Datei ausführen / Verzeichnis betreten

Berechtigungen mit ls -l verstehen

Aufbau der Berechtigungsdarstellung:

```
ls -l datei.txt
# -rwxr--r-- 1 user group 1234 Jan 15 10:30 datei.txt
# ||||| |
# ||||| |____ Others: read only
# ||||| |____ Group: read only
# |LL|____ Owner: read, write, execute
# |____ File type: - (regular file)
```

Die ersten 10 Zeichen erklärt:

- Position 1: Dateityp (`-`, `d`, `l`, etc.)
- Position 2-4: Owner-Berechtigungen (`rwx`)
- Position 5-7: Group-Berechtigungen (`r--`)
- Position 8-10: Others-Berechtigungen (`r--`)

Besitzer und Gruppen anzeigen

Wem gehört welche Datei?

```
# Detaillierte Anzeige mit Besitzer
ls -l
# -rw-r--r-- 1 janosch users 42 Jan 15 10:30 datei.txt
#           ^      ^
#           Owner  Group
```

Berechtigungen ändern

chmod - Change Mode

```
# Symbolische Notation
chmod u+x datei.txt      # Owner: Execute-Berechtigung hinzufügen
chmod g-w datei.txt      # Group: Write-Berechtigung entfernen
chmod o=r datei.txt      # Others: Nur Read-Berechtigung

# Oktale Notation
chmod 755 datei.txt      # rwxr-xr-x
chmod 644 datei.txt      # rw-r--r--
chmod 600 datei.txt      # rw-----
```

Symbolische chmod-Notation

Flexible Berechtigungsänderungen:

```
# Wer (Who):  
# u = user (owner), g = group, o = others, a = all  
  
# Was (What):  
# + = hinzufügen, - = entfernen, = = setzen  
  
# Welche (Which):  
# r = read, w = write, x = execute  
  
# Beispiele:  
chmod u+rw datei.txt      # Owner: Read und Write hinzufügen  
chmod go-rwx datei.txt    # Group und Others: Alle Rechte entfernen  
chmod a=r datei.txt       # Alle: Nur Read-Berechtigung
```

Oktale chmod-Notation

Binär	Oktal	Symbolisch	Bedeutung
000	0	---	Keine Rechte
001	1	--x	Nur Execute
010	2	-w-	Nur Write
011	3	-wx	Write + Execute
100	4	r--	Nur Read
101	5	r-x	Read + Execute
110	6	rw-	Read + Write
111	7	rwX	Alle Rechte

Häufige chmod-Muster

```
# Dateien
chmod 644 datei.txt      # rw-r--r-- (Standard für Dateien)
chmod 600 datei.txt      # rw----- (Nur Owner)
chmod 755 skript.sh      # rwxr-xr-x (Ausführbares Skript)

# Verzeichnisse
chmod 755 verzeichnis/   # rwxr-xr-x (Standard für Verzeichnisse)
chmod 750 verzeichnis/   # rwxr-x--- (Gruppe kann betreten)
chmod 700 verzeichnis/   # rwx----- (Nur Owner)

# Rekursiv für Verzeichnisbäume
chmod -R 755 /pfad/zum/verzeichnis/
```

Verzeichnis-Berechtigungen verstehen

- **Read (r):** Verzeichnisinhalt auflisten (`ls`)
- **Write (w):** Dateien erstellen/löschen im Verzeichnis
- **Execute (x):** Verzeichnis betreten (`cd`)

```
# Verschiedene Kombinationen testen
mkdir test-dir
echo "test" > test-dir/datei.txt

# Nur Read: Inhalt sehen aber nicht betreten
chmod 444 test-dir
ls -f test-dir          # ✓ Funktioniert
cat test-dir/datei.txt  # ✗ Fehler

# Nur Execute: betreten aber nicht auflisten
chmod 111 test-dir
ls test-dir             # ✗ Fehler
cd test-dir && pwd       # ✓ Funktioniert
```

chown - Besitzer ändern

chown - Change Owner

```
# Nur Benutzer ändern
sudo chown neuer-user datei.txt

# Nur Gruppe ändern
sudo chown :neue-gruppe datei.txt

# Benutzer und Gruppe ändern
sudo chown neuer-user:neue-gruppe datei.txt

# Rekursiv für Verzeichnisse
sudo chown -R user:group verzeichnis/
```

Zur Kapitelübersicht

- Vorheriges Kapitel: [File System](#)
- Nächstes Kapitel: [SSH Grundlagen](#)