3. Dateien suchen, anzeigen und bearbeiten

3.1 Dateien suchen, anzeigen und bearbeiten

In diesem Kapitel geht es um den praktischen Umgang mit Dateien unter Linux.

Dazu gehören:

- Dateien anzeigen
- Dateien erstellen
- Dateien kopieren
- Dateien verschieben
- Dateien löschen
- Inhalte lesen
- Inhalte filtern
- Dateien suchen
- einfache Textbearbeitung

Für Fachinformatiker Systemintegration ist das wichtig, weil viele Linux-Aufgaben direkt mit Dateien, Konfigurationen, Logs und Skripten zu tun haben.

Merksatz:

Unter Linux ist sehr vieles eine Datei:
Konfigurationen,
Logs,
Geräteinformationen
und Skripte.

Lernziele

Nach dieser Seite solltest du erklären und anwenden können:

- wie man Dateien und Verzeichnisse auflistet
- wie man Dateiinhalte anzeigt
- wie man Dateien erstellt
- wie man Dateien kopiert,
  verschiebt

und löscht
- wie man nach Dateien sucht
- wie man Inhalte in Dateien findet
- wofür cat,
less,
head,
tail,
grep,
find
und nano genutzt werden
- welche typischen Fehler beim Umgang mit Dateien auftreten

Grundidee

Viele administrative Aufgaben unter Linux bestehen daraus, Dateien zu lesen, zu ändern oder zu prüfen.

Beispiele:

- Konfigurationsdatei unter /etc bearbeiten
- Logdatei unter /var/log auswerten
- Skript im Home-Verzeichnis erstellen
- Backup-Dateien prüfen
- Dienstkonfiguration vergleichen
- Text in Dateien suchen
- alte Dateien finden und löschen

Wichtig:

Vor Änderungen an wichtigen Konfigurationsdateien sollte man eine Sicherungskopie erstellen.

Merksatz:

Erst lesen,
dann sichern,
dann ändern.

Dateien und Verzeichnisse anzeigen

Der wichtigste Befehl zum Anzeigen von Verzeichnisinhalten ist:

```
ls
```

Beispiele:

```
ls

ls -l

ls -a

ls -la
```

Bedeutung:

Befehl	Bedeutung
ls	Inhalt des aktuellen Verzeichnisses anzeigen
ls -l	ausführliche Liste anzeigen
ls -a	auch versteckte Dateien anzeigen
ls -la	ausführlich und mit versteckten Dateien
ls /etc	Inhalt von /etc anzeigen
ls -lh	Größen menschenlesbar anzeigen

Merksatz:

```
ls zeigt Dateien und Verzeichnisse an.
```

Ausgabe von ls -l verstehen

Beispiel:

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

Bedeutung:

Teil	Bedeutung

-rw-r--r--	Dateityp und Rechte
1	Anzahl der Links
felix	Besitzer
users	Gruppe
1200	Dateigröße in Byte
Jul 08 12:30	Änderungsdatum
beispiel.txt	Dateiname

Der erste Buchstabe zeigt den Dateityp:

Zeichen	Bedeutung
-	normale Datei
d	Verzeichnis
l	symbolischer Link
c	Zeichen-Gerät
b	Block-Gerät
s	Socket
p	Pipe

Merksatz:

```
ls -l zeigt Rechte,
Besitzer,
Größe,
Datum
und Dateinamen.
```

Dateiinhalte mit cat anzeigen

cat zeigt den Inhalt einer Datei direkt im Terminal an.

Beispiel:

```
cat datei.txt
```

Typische Nutzung:

kleine Textdateien anzeigen

Konfigurationsdateien kurz prüfen

Inhalt in Skripten ausgeben

Wichtig:

cat ist unpraktisch bei sehr langen Dateien,
weil der Inhalt direkt durchläuft.

Merksatz:

cat zeigt kleine Dateien direkt an.

Dateiinhalte mit less anzeigen

less zeigt Dateien seitenweise an.

Beispiel:

```
less /var/log/syslog
```

Navigation in less:

Taste	Bedeutung
Leertaste	eine Seite weiter
b	eine Seite zurück
Pfeiltasten	zeilenweise bewegen
/suchwort	innerhalb der Datei suchen
n	nächster Treffer
q	less verlassen

Vorteil:

less eignet sich gut für lange Dateien und Logs.

Merksatz:

```
less ist besser für lange Dateien.
```

Anfang einer Datei anzeigen

Mit head zeigt man den Anfang einer Datei.

Beispiel:

```
head datei.txt
```

Standardmäßig werden die ersten 10 Zeilen angezeigt.

Mehr Zeilen anzeigen:

```
head -n 20 datei.txt
```

Typische Nutzung:

```
Anfang einer Logdatei prüfen

CSV-Datei kurz ansehen

Kopfzeilen prüfen
```

Merksatz:

```
head zeigt den Anfang.
```

Ende einer Datei anzeigen

Mit tail zeigt man das Ende einer Datei.

Beispiel:

```
tail datei.txt
```

Standardmäßig werden die letzten 10 Zeilen angezeigt.

Mehr Zeilen anzeigen:

```
tail -n 50 datei.txt
```

Live mitlesen:

```
tail -f /var/log/syslog
```

Typische Nutzung:

```
aktuelle Logeinträge verfolgen

Dienstfehler live beobachten

neue Einträge prüfen
```

Merksatz:

```
tail zeigt das Ende.
tail -f liest live mit.
```

Dateien erstellen mit touch

touch erstellt eine leere Datei, wenn sie noch nicht existiert.

Beispiel:

```
touch notiz.txt
```

Wenn die Datei bereits existiert, wird der Zeitstempel aktualisiert.

Typische Nutzung:

```
leere Datei anlegen

Testdatei erstellen
```

Zeitstempel aktualisieren

Merksatz:

touch erstellt leere Dateien oder aktualisiert Zeitstempel.

Verzeichnisse erstellen mit mkdir

Mit mkdir erstellt man Verzeichnisse.

Beispiel:

mkdir testordner

Mehrere Ebenen erstellen:

mkdir -p projekt/logs/archiv

Bedeutung:

Befehl	Bedeutung
mkdir test	Verzeichnis test erstellen
mkdir -p a/b/c	komplette Verzeichnisstruktur erstellen

Merksatz:

mkdir erstellt Verzeichnisse.
mkdir -p erstellt auch fehlende Zwischenordner.

Dateien kopieren mit cp

Mit cp kopiert man Dateien.

Beispiele:

cp quelle.txt ziel.txt

```
cp datei.txt /tmp/

cp /etc/hosts ~/hosts.backup
```

Verzeichnisse rekursiv kopieren:

```
cp -r ordner1 ordner2
```

Wichtige Optionen:

Option	Bedeutung
-r	rekursiv, für Verzeichnisse
-i	vor Überschreiben nachfragen
-v	zeigt ausgeführte Aktionen
-p	erhält Zeitstempel und Rechte möglichst bei

Merksatz:

```
cp kopiert.
Für Verzeichnisse braucht man meist cp -r.
```

Dateien verschieben oder umbenennen mit mv

Mit mv verschiebt oder benennt man Dateien um.

Umbenennen:

```
mv alt.txt neu.txt
```

Verschieben:

```
mv datei.txt /tmp/
```

Verzeichnis verschieben:

```
mv projekt /tmp/
```

Wichtig:

```
mv kann bestehende Dateien überschreiben,
wenn man nicht aufpasst.
```

Sicherer:

```
mv -i alt.txt neu.txt
```

Merksatz:

```
mv verschiebt und benennt um.
```

Dateien löschen mit rm

Mit rm löscht man Dateien.

Beispiel:

```
rm datei.txt
```

Mit Nachfrage:

```
rm -i datei.txt
```

Verzeichnis mit Inhalt löschen:

```
rm -r ordner
```

Sehr vorsichtig verwenden:

```
rm -rf ordner
```

Bedeutung:

Option	Bedeutung
-r	rekursiv, auch Unterordner
-f	force, keine Rückfrage
-i	interaktiv, mit Nachfrage

Wichtig:

```
rm löscht normalerweise nicht in einen Papierkorb.
Gelöschte Dateien sind nicht einfach wiederherstellbar.
```

Merksatz:

```
rm löscht direkt.
rm -rf nur sehr bewusst verwenden.
```

Leere Verzeichnisse löschen mit rmdir

rmdir löscht nur leere Verzeichnisse.

Beispiel:

```
rmdir leerer_ordner
```

Wenn das Verzeichnis nicht leer ist, schlägt der Befehl fehl.

Merksatz:

```
rmdir löscht nur leere Verzeichnisse.
```

Dateien mit nano bearbeiten

nano ist ein einfacher Texteditor im Terminal.

Datei öffnen oder erstellen:

```
nano datei.txt
```

Wichtige Tastenkombinationen:

Tastenkombination	Bedeutung
Strg + O	speichern
Enter	Dateiname bestätigen
Strg + X	nano verlassen

Tastenkombination	Bedeutung
Strg + W	suchen
Strg + K	Zeile ausschneiden
Strg + U	Zeile einfügen

Typische Nutzung:

```
Konfigurationsdateien bearbeiten

Notizen erstellen

kleine Skripte schreiben
```

Merksatz:

```
nano ist ein einfacher Terminal-Editor.
```

Dateien sicher bearbeiten

Vor Änderungen an wichtigen Dateien sollte man eine Sicherung erstellen.

Beispiel:

```
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
```

Danach bearbeiten:

```
nano /etc/ssh/sshd_config
```

Vorteil:

```
Bei Fehlern kann man die alte Version wiederherstellen.
```

Beispiel:

```
cp /etc/ssh/sshd_config.bak /etc/ssh/sshd_config
```

Merksatz:

```
Vor Konfigurationsänderung Backup-Datei erstellen.
```

In Dateien suchen mit grep

grep sucht Text in Dateien.

Beispiel:

```
grep "error" logfile.txt
```

Groß- und Kleinschreibung ignorieren:

```
grep -i "error" logfile.txt
```

Zeilennummern anzeigen:

```
grep -n "error" logfile.txt
```

Rekursiv in Verzeichnissen suchen:

```
grep -r "PermitRootLogin" /etc/ssh
```

Wichtige Optionen:

Option	Bedeutung
-i	Groß-/Kleinschreibung ignorieren
-n	Zeilennummer anzeigen
-r	rekursiv in Verzeichnissen suchen
-v	Treffer umkehren, also nicht passende Zeilen
-l	nur Dateinamen mit Treffern anzeigen

Merksatz:

```
grep sucht Textmuster in Dateien.
```

Typische grep-Beispiele

Fehler in Logdatei suchen:

```
grep -i "error" /var/log/syslog
```

SSH-Konfiguration prüfen:

```
grep "PermitRootLogin" /etc/ssh/sshd_config
```

Alle Zeilen ohne Kommentar anzeigen:

```
grep -v "^#" datei.conf
```

Mehrere Treffer mit Zeilennummern:

```
grep -n "root" /etc/passwd
```

Merksatz:

```
grep ist eines der wichtigsten Werkzeuge für Logs und Konfigurationen.
```

Dateien suchen mit find

find sucht Dateien und Verzeichnisse im Dateisystem.

Beispiele:

```
find /home -name "datei.txt"

find /var/log -name "*.log"

find /etc -name "*ssh*"
```

Bedeutung:

Befehl	Bedeutung
find /home -name "test.txt"	sucht test.txt unter /home

Befehl	Bedeutung
find . -name "*.txt"	sucht .txt-Dateien ab aktuellem Verzeichnis
find /var/log -type f	sucht normale Dateien unter /var/log
find /var/log -type d	sucht Verzeichnisse unter /var/log

Merksatz:

```
find sucht Dateien nach Kriterien.
```

---

find nach Dateityp

Mit -type kann man Dateitypen einschränken.

Beispiele:

```
find /home -type f

find /home -type d

find /home -type l
```

Bedeutung:

Option	Bedeutung
-type f	normale Dateien
-type d	Verzeichnisse
-type l	symbolische Links

Merksatz:

```
find -type f sucht Dateien.
find -type d sucht Verzeichnisse.
```

---

find nach Größe

Dateien größer als 100 MB finden:

```
find /home -type f -size +100M
```

Dateien kleiner als 1 MB finden:

```
find /home -type f -size -1M
```

Dateien genau 10 KB:

```
find /home -type f -size 10k
```

Merksatz:

```
find kann auch nach Dateigröße suchen.
```

---

find nach Zeit

Dateien suchen, die in den letzten 7 Tagen geändert wurden:

```
find /home -type f -mtime -7
```

Dateien suchen, die älter als 30 Tage sind:

```
find /tmp -type f -mtime +30
```

Bedeutung:

Ausdruck	Bedeutung
-mtime -7	jünger als 7 Tage
-mtime +30	älter als 30 Tage
-mtime 1	ungefähr vor einem Tag geändert

Merksatz:

```
find kann alte oder neue Dateien finden.
```

---

which und whereis

which zeigt, welcher Programmpfad beim Ausführen verwendet wird.

Beispiel:

```
which ssh
```

Mögliche Ausgabe:

```
/usr/bin/ssh
```

whereis zeigt weitere Orte zu einem Programm an.

Beispiel:

```
whereis ssh
```

Merksatz:

```
which zeigt,
welches Programm ausgeführt wird.
```

Dateiinformationen mit file anzeigen

file erkennt den Dateityp anhand des Inhalts.

Beispiel:

```
file dokument.txt

file script.sh

file bild.png
```

Nützlich, wenn die Dateiendung unklar oder falsch ist.

Merksatz:

```
file erkennt Dateitypen anhand des Inhalts.
```

Dateigrößen anzeigen

Mit ls -lh werden Größen besser lesbar angezeigt.

Beispiel:

```
ls -lh
```

Mit du kann man Speicherverbrauch anzeigen.

Beispiel:

```
du -sh ordner
```

Bedeutung:

Befehl	Bedeutung
ls -lh	Dateigrößen lesbar anzeigen
du -sh ordner	Gesamtgröße eines Ordners anzeigen
du -h	Größen rekursiv anzeigen

Merksatz:

```
ls zeigt Dateigröße.
du zeigt Speicherverbrauch.
```

Dateien zählen

Dateien in einem Verzeichnis zählen:

```
ls | wc -l
```

Alle normalen Dateien unter einem Verzeichnis zählen:

```
find /home/felix -type f | wc -l
```

Bedeutung:

Befehl	Zweck

wc -l	Zeilen zählen
find ... | wc -l	gefundene Einträge zählen

Merksatz:

```
wc -l zählt Zeilen.
```

Ausgaben mit Platzhaltern

Die Shell kann Platzhalter verwenden.

Platzhalter	Bedeutung
*	beliebige Zeichen
?	genau ein beliebiges Zeichen
[abc]	eines der Zeichen a, b oder c

Beispiele:

```
ls *.txt

rm *.tmp

cp *.conf backup/
```

Wichtig:

```
Platzhalter werden von der Shell erweitert,
bevor der Befehl ausgeführt wird.
```

Merksatz:

```
* steht für beliebige Zeichen.
```

Achtung bei rm und Platzhaltern

Beispiel:

```
rm *.tmp
```

löscht alle Dateien, die auf .tmp enden.

Gefährlich:

```
rm *
```

löscht alle passenden Dateien im aktuellen Verzeichnis.

Noch gefährlicher:

```
rm -rf *
```

löscht rekursiv alles Passende im aktuellen Verzeichnis.

Merksatz:

```
Vor rm mit Platzhaltern immer prüfen,
was betroffen ist.
```

Sichere Arbeitsweise

Vor riskanten Befehlen:

```
pwd

ls

ls -la

echo *.tmp

dann erst löschen oder verschieben
```

Beispiel:

```
echo *.tmp
```

zeigt, welche Dateien vom Platzhalter betroffen wären.

Merksatz:

> Erst anzeigen,
> dann löschen.

Typische Datei-Aufgaben im Admin-Alltag

Aufgabe	Möglicher Befehl
aktuelle Position anzeigen	pwd
Dateien anzeigen	ls -la
Datei lesen	cat datei.txt
lange Datei lesen	less datei.txt
Log live verfolgen	tail -f logfile
Datei erstellen	touch datei.txt
Verzeichnis erstellen	mkdir ordner
Datei kopieren	cp quelle ziel
Verzeichnis kopieren	cp -r quelle ziel
Datei verschieben	mv quelle ziel
Datei löschen	rm datei
leeres Verzeichnis löschen	rmdir ordner
Text suchen	grep "text" datei
Datei suchen	find /pfad -name "datei"
Datei bearbeiten	nano datei

Typische Fehlerbilder

Fehlerbild	Wahrscheinliche Ursache
Datei nicht gefunden	falscher Pfad oder falscher Name
Permission denied	fehlende Rechte
No such file or directory	Datei oder Verzeichnis existiert nicht
Is a directory	Befehl erwartet Datei, Ziel ist Verzeichnis
Not a directory	Pfadbestandteil ist keine Ordnerstruktur

Fehlerbild	Wahrscheinliche Ursache
Datei leer nach Umleitung	falscher Einsatz von >
rm löscht zu viel	Platzhalter oder falscher Pfad
grep findet nichts	Groß-/Kleinschreibung oder falsches Muster
find findet nichts	falscher Startpfad oder Suchkriterium

Permission denied

Die Meldung:

```
Permission denied
```

bedeutet:

```
Zugriff verweigert
```

Mögliche Ursachen:

```
fehlende Leserechte

fehlende Schreibrechte

fehlende Ausführrechte

Datei gehört anderem Benutzer

Verzeichnisrechte fehlen

administrative Rechte nötig
```

Nicht immer ist sudo die beste Lösung.

Besser zuerst prüfen:

```
ls -l datei

ls -ld verzeichnis
```

Merksatz:

```
Permission denied bedeutet Rechteproblem.
```

No such file or directory

Die Meldung:

```
No such file or directory
```

bedeutet:

```
Datei oder Verzeichnis wurde nicht gefunden.
```

Mögliche Ursachen:

```
Tippfehler

falscher Pfad

falsche Groß-/Kleinschreibung

Datei wurde gelöscht

relativer Pfad falsch verstanden
```

Prüfen:

```
pwd

ls

ls -la
```

Merksatz:

```
Linux unterscheidet Groß- und Kleinschreibung.
```

Vor Änderungen an Konfigurationsdateien

Sicheres Vorgehen:

```
1. Datei ansehen

2. Backup erstellen

3. Datei bearbeiten

4. Syntax prüfen,
   falls möglich

5. Dienst neu laden oder neu starten

6. Status prüfen

7. Logs prüfen
```

Beispiel:

```
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak

nano /etc/ssh/sshd_config

systemctl restart ssh

systemctl status ssh
```

Merksatz:

```
Konfiguration nie blind ändern.
```

Wichtige Befehle dieser Seite

Befehl	Zweck
ls	Dateien anzeigen
cat	Datei direkt ausgeben

Befehl	Zweck
less	Datei seitenweise lesen
head	Anfang anzeigen
tail	Ende anzeigen
tail -f	live mitlesen
touch	leere Datei erstellen
mkdir	Verzeichnis erstellen
cp	kopieren
mv	verschieben oder umbenennen
rm	löschen
rmdir	leeres Verzeichnis löschen
nano	Datei bearbeiten
grep	Text suchen
find	Dateien suchen
file	Dateityp erkennen
which	Programmpfad anzeigen
du	Speicherverbrauch anzeigen
wc	zählen

Typische Prüfungsfragen

Frage	Kurzantwort
Wofür ist ls?	Verzeichnisinhalt anzeigen
Wofür ist cat?	Dateiinhalt direkt anzeigen
Wofür ist less?	lange Dateien seitenweise anzeigen
Wofür ist tail -f?	Datei live mitlesen
Wofür ist touch?	leere Datei erstellen oder Zeitstempel ändern
Wofür ist cp?	kopieren
Wofür ist mv?	verschieben oder umbenennen
Wofür ist rm?	löschen
Wofür ist grep?	Text in Dateien suchen
Wofür ist find?	Dateien im Dateisystem suchen
Wofür ist nano?	Textdateien im Terminal bearbeiten

Frage	Kurzantwort
Was bedeutet Permission denied?	fehlende Rechte
Was bedeutet No such file or directory?	Datei oder Pfad existiert nicht

---

Typische Prüfungsfallen

Falle	Richtig denken
cat für riesige Logs nutzen	besser less oder tail verwenden
rm wie Papierkorb verstehen	rm löscht direkt
cp ohne -r bei Verzeichnissen	Verzeichnisse rekursiv kopieren
mv nur als Verschieben verstehen	mv benennt auch um
grep ohne -i bei unbekannter Schreibweise	-i ignoriert Groß-/Kleinschreibung
find im falschen Startpfad	Suchpfad bewusst wählen
sudo sofort verwenden	zuerst Rechte und Pfad prüfen
Konfigurationsdatei ohne Backup ändern	vorher Kopie erstellen
Platzhalter bei rm unterschätzen	vorher mit echo prüfen
relative Pfade falsch verstehen	pwd prüfen

IHK-sichere Kurzformulierung

Unter Linux werden Dateien und Verzeichnisse häufig über die Shell verwaltet. Mit ls werden Verzeichnisinhalte angezeigt, mit cat, less, head und tail werden Dateiinhalte gelesen. tail -f eignet sich besonders zum Live-Mitlesen von Logdateien. Dateien werden mit touch erstellt, mit cp kopiert, mit mv verschoben oder umbenannt und mit rm gelöscht. grep sucht Textmuster in Dateien, während find Dateien und Verzeichnisse anhand von Kriterien wie Name, Typ, Größe oder Änderungszeit sucht. Vor Änderungen an wichtigen Konfigurationsdateien sollte eine Sicherungskopie erstellt werden. Fehlermeldungen wie Permission denied deuten auf Rechteprobleme hin, während No such file or directory auf einen falschen oder nicht vorhandenen Pfad hinweist.

Merksätze

ls zeigt Dateien und Verzeichnisse.

ls -l zeigt Details.

ls -a zeigt versteckte Dateien.

cat zeigt kleine Dateien direkt.

less ist besser für lange Dateien.

head zeigt den Anfang.

tail zeigt das Ende.

tail -f liest live mit.

touch erstellt leere Dateien.

mkdir erstellt Verzeichnisse.

mkdir -p erstellt Zwischenordner mit.

cp kopiert.

cp -r kopiert Verzeichnisse.

mv verschiebt und benennt um.

rm löscht direkt.

rmdir löscht leere Verzeichnisse.

nano bearbeitet Textdateien.

grep sucht Text.

find sucht Dateien.

file erkennt Dateitypen.

which zeigt den Programmpfad.

du zeigt Speicherverbrauch.

wc -l zählt Zeilen.

Permission denied bedeutet Rechteproblem.

No such file or directory bedeutet:
Datei oder Pfad existiert nicht.

Vor Konfigurationsänderungen Backup erstellen.

Erst anzeigen,
dann löschen.

Platzhalter bei rm sehr vorsichtig verwenden.

Linux unterscheidet Groß- und Kleinschreibung.