Kapitel 4

Dateien

Lernziele
Nach Abschluss des Kapitels

- können Sie mit Dateien unter Linux umgehen,
- kennen Sie die wichtigsten Systemkommandos,
- können Sie Textdateien einsehen und bearbeiten.

4.1 Einleitung

Die Verarbeitung von Daten ist der zentrale Punkt bei der Nutzung von Rechnern. Im Gegensatz zu früher, wo Daten im laufenden Betrieb ein und ausgegeben und nur für Rechenschritte im Arbeitsspeicher vorgehalten wurden, ist auf heutigen Rechnersystemen die Verwaltung von Daten auf Speichermedien in Form von Dateien eine wesentliche Aufgabe geworden. Betriebssysteme werden deshalb mit einem umfangreichen Satz an Systemprogrammen ausgeliefert, die bei der Verwaltung und Bearbeitung von Dateien unterstützen. In diesem Kapitel wird die Verarbeitung von Dateien auf der Ebene der Shell und Systemprogramme fokusiert. In einem folgenden Kapitel wird dann auch die Frage der Speicherung von Dateien in Dateisystemen auf Speichermedien thematisiert.

4.2 Navigation im Verzeichnisbaum

Im Dateisystem von Unix wird alles als Datei behandelt. Selbst ein Verzeichnis ist eigentlich eine Datei, wie wir später sehen werden.

Merken Sie sich also den Grundsatz für Unix:

Alles ist Datei!

Bei der Arbeit mit dem Prompt auf der Konsole befindet man sich immer in einem Verzeichnis.

Wird kein alternativer Verzeichnispfad für einen Befehl angegeben, so beziehen sich alle Befehle immer auf das aktuelle Verzeichnis.

Um sich das aktuelle Verzeichnis anzeigen lassen zu können, verwenden wir den Befehl:

```
#pwd

```

Der Inhalt des aktuellen Verzeichnisses lässt sich mit dem Befehl

```
ls

```

darstellen. Dies steht als Kurform für list. In Unix werden häufig mnemographie Kürzel verwendet, um die Schreibarbeit zur reduzieren, aber trotzdem den Anwender zu unterstützen.

In der Auflistung findet man alle Dateien und Verzeichnisse, die sich im aktuellen Verzeichnis befinden. In vielen Unix-Systemen werden die Datei- und Verzeichnistypen farblich in der Anzeige unterschieden. Blau steht hierbei z.B. für Verzeichnisse und grün für ausführbare Programmdateien.

Mit dem Befehl für Change Directory unter Angabe des Verzeichnisses kann man nun in ein anderes Verzeichnis wechseln.

```
cd <Verzeichnisname>
ls

```

Nach Ausführung des cd-Befehls erfolgt dabei keine Ausgabe oder Rückgabe eines Wertes. Es wird einfach nur das Verzeichnis gewechselt. Wird ein Verzeichnisname aus dem aktuellen Verzeichnis ausgewählt, springt man mit dem cd-Befehl direkt in der Hierarchie der Verzeichnisse um eine Stufe nach unten.

Unix-Systeme sind case-sensitive. Es wird genau zwischen Groß- und Kleinschreibung bei Dateien und Verzeichnissen unterschieden. "Testverzeichnis" nicht gleich "testverzeichnis"

Mit cd kann man aber auch kreuz und quer durch den Verzeichnisbaum springen, wenn man die genaue Adresse kennt.

```
cd /usr

ls

```

Mit der Adressangabe /usr springt man direkt in das Verzeichnis usr, das sich in der ersten Ebene nach dem Basisverzeichnis befindet.

Bei der konventionellen Ausführung des ls-Befehls werden keine versteckten Dateien angezeigt. Durch Optionen können Befehle erweitert werden. Für den ls-Befehl ist die wichtige Optionen -a.

```
ls -a

```

Durch den Bindestrich wird dem Befehl angekündigt, dass nun eine oder mehrere Optionen folgen. Die Option a steht dabei für all und führt dazu, dass der ls-Befehl auch versteckte Dateien und Verzeichnisse aufführt. Diese sind an einem Punkt zu Beginn des Dateinamens zu erkennen.

Zwei wichtige Verzeichnisse verstecken sich hinter den Dateinamen:

```
ls -a
. entspricht dem aktuellen Verzeichnis
.. entspricht dem untergeordneten Verzeichnis
```

Diese beiden Verzeichnisse sind Verweise im Verzeichnisbaum auf das aktuelle Verzeichnis (.) sowie das hierarchisch untergeordnete Verzeichnis (..).

Um nun aus /usr wieder in das Hauptverzeichnis / wechseln zu können, reicht also ein einfaches

```
cd ..
ls

```

Eine Alternative hierzu wäre der Verzeichnisname des Hauptverzeichnisses /

```
 cd /


```

Der Verzeichnisname des eigenen Homeverzeichnisses, da wo wir eigentlich hergekommen sind, lautet .

```
cd ~ (Alt Gr + +)


```

bringt uns also wieder zurück nachhause. Um in das eigene Homeverzeichnis zu wechseln, kann das -Symbol auch weggelassen werden.

```
1 cd /home/<benutzername>
2
3
```

Übung

1. Überprüfen Sie, ob Sie auf Konsole 1 (tty1) sind bzw. wechseln Sie dorthin (<Strg> + <Alt> + <F1>).
2. Lassen Sie sich mit pwd das aktuelle Verzeichnis anzeigen.
3. Wechseln Sie ggf. mit cd in das Homeverzeichnis.
4. Lassen Sie sich mit ls und ls -a den Inhalt des Homeverzeichnisses anzeigen und überprüfen Sie den Unterschied zwischen beiden Kommandos. Sollte die Ausgabe zu lange sein, dann können Sie mit der Tastenkombination Shift + Bild ↑ den oberen Bereich der Ausgabe ansehen.
5. Wechseln Sie mit cd <Verzeichnisname> in ein übergeordnetes Verzeichnis.
6. Lassen Sie sich den Verzeichnisinhalt mit ls bzw. ls -a anzeigen.
7. Wechseln Sie mit cd .. zurück in das Homeverzeichnis
8. Überprüfen Sie den Wechsel mit pwd bzw. ls.
9. Wechseln Sie mit cd /etc in das Konfigurationsverzeichnis.
10. lassen Sie sich den Verzeichnisinhalt anzeigen. In /etc werden wir später alle wichtigen Konfigurationen des Systems auf Basis von Textdateien als Steuerungsdateien vornehmen.
11. Wechseln Sie mit cd .. in das Hauptverzeichnis.
12. Sehen Sie sich die grundlegende Verzeichnisstruktur von Unix an, die Sie in fast allen Unix-Systemen finden.
13. Wechseln Sie mit cd zurück in Ihr Homeverzeichnis.

4.3 Verzeichnisbaum

Es gibt bestimmte Verzeichnisse, die man auf jeder UNIX-Maschine findet.

4.3.1 Aufbau der Verzeichnisstruktur

Unix-Verzeichnisse folgen in der Regel einer generellen Konvention und sind, bis auf einige kleinere Ausnahmen und Abweichungen, nach dem gleichen Schema aufgebaut.

Der Unix-Verzeichnisbaum startet immer mit dem Root-Verzeichnis /

Dieses Root-Verzeichnis ist bereits selbst ein Verzeichnis (in der Windowswelt auch Ordner genannt) und kann weitere Verzeichniss und Dateien beinhalten. Am Anfang eines Verzeichnispfades ist der Schrägstrich / immer das Symbol bzw. der Name für das Rootverzeichnis.

An anderen Stellen im Verzeichnispfad ist der Schrägstrich / als Trennzeichen zwischen einzelnen Hierarchieebenen der Verzeichnisse reserviert.

Achten Sie übrigens darauf, dass der Schrägstrich im Unix-System immer ein Slash / ist und nicht, wie unter Windows üblich, ein Backslash
.

Die neuen wichtigsten Verzeichnisse lauten wie folgt:

/etc

/bin

/lib

/tmp

/usr

/var

/home

/opt

4.3.2 Bedeutung der Hauptverzeichnisse

/etc

- Konfigurationsdateien (ggf. in Unterordnern)
- Start-Skripte von Unix / Linux
- Textdateien mit Konfiguratitonsparametern (Bsp. /etc/samba/smb.conf)

/bin

- Programme (Unix-Befehle z. B. $ls, cd, ...)
- Minimalset an Programmen für ein Unix/Linux System
- Verzeichnis sollte auf der Bootpartition liegen

/lib

- Ablage aller dynamischen Bibliotheken der Programma aus /bin
- Dynamische Bibliotheken sind Dateien mit Programmteilen die von mehreren, anderen Programmen genutzt werden
- Verzeichnis sollte auf der Bootpartition liegen

/tmp

- Ablage von temporären Dateien
- für jeden Benutzer schreib- und lesbar
- immer an der selben Stelle zu finden
- Inhalt wird von manchen System regelmäßig gelöscht

/usr

- Anwenderprogramme abgelegt (/usr/bin)
- zugehörige Bibliotheken (/usr/lib)
- zugehörige Manpages (/usr/man)

/var

- Dateien, die vom System oder von Anwenderprogrammen verändert werden (z. B. Protokolldateien, Spooling (Druckdienste))
- Auf anderen Distributionen sind diese Dateien ggf. auch an anderen Stellen abgelegt

/home

- Basisverzeichnis in dem für jeden Nutzer ein eigenes (persönliches) Unterverzeichnis abgelegt wird
- Name des Unterverz. = Name des Benutzers
- Kurzadressierung: ~ (z. B. cd ~)

/opt

- Ablegen größerer Programmpakete (z. B. OpenOffice, Acrobat Reader, StarUML, ...)
- Optionaler Installationspunkt

4.4 Logout

Achten Sie darauf, dass Sie sich nach Abschluss der Arbeiten von allen Konsolen abmelden. Der Logout an der Konsole erfolgt per

```
1 #exit
```

oder

```
1 #logout
```

4.5 Grundlagen

In einem Unix-System ist fast alles eine Datei. Der Zugriff auf Geräte erfolgt über Gerätedateien (special files). Verzeichnisse sind eine Sonderform der Datei und auch Links auf Dateien werden als Datei realisiert. Dazu aber später mehr. Aber auch in anderen Systemen ist das Konzept der Datei geläufig und wird technisch ähnlich realisiert. Anhand einiger kleiner Arbeitsschritte wollen wir uns die Arbeit mit wichtigen Systemprogrammen ansehen, die die Ver- und Bearbeitung von Dateien erlauben.

Einen Überblick über Dateien können wir uns mit dem Befehl ls verschaffen. Angezeigt werden alle Dateien, die sich innerhalb des aktuellen Verzeichnisses befinden:

```
$ ls -la
```

Die Dateioption -la führt dazu, dass auch sogenannte versteckte Dateien (haben als erste Zeichen den . im Dateinamen) sowie alle Dateieigenschaften aufgelistet werden. Die farbliche Kennzeichnung weist auf den Typ der Datei (Verzeichnis, ausführbares Programm, Link, ...). Im Gegensatz zu Windows kommt der Dateiendung auf Ebene des Betriebssystems keine Bedeutung bei.

Die -la Option gibt dabei Informationen zu:

```
1 drwxr-xr-x  2 wdorner wdorner      4096 Aug 11 12:56 test
```

- den Rechten die unterschiedliche Nutzer an Dateien haben
- der Anzahl an Dateien innerhalb eines Verzeichnisses bzw. der Links auf die Datei
- dem Eigentümer und der Gruppe der die Datei "gehört"
- der Größe der Datei
- Datum und Uhrzeit der letzten Änderung
- Dateinamen

Mit Dateien wollen wir nun in einem eigenen Verzeichnis arbeiten, das wir uns zu diesem Zweck mit dem Systemprogramm mkdir anlegen:

```
1 $mkdir testverzeichnis
2 $ cd testverzeichnis
```

Um eine Datei zur Bearbeitung zu haben, kopieren wir uns eine erste Datei in unser Verzeichnis:

Erforderliche Befehle:

- cp <Pfad>/<Dateinamen> kopiert eine Datei in das aktuelle Verzeichnis
- cp <Dateiname> <Dateiname2> erstellt eine Dateikopie
- mv <Dateiname> <Pfad> verschiebt eine Datei in ein Verzeichnis
- mv <Dateiname> <Dateiname2> benennt eine Datei um
- rm <Dateiname> löscht eine Datei
- rmdir <Verzeichnisname> löscht ein leeres Verzeichnis
- rm -R <Verzeichnisname> rekursives löschen eines vollen Verzeichnisses

(Übung)

Hinweise zu cp:

- Kopieren von Verzeichnisbäumen rekursiv mit $cp -R b zw. $cp -r
- Neue Datei wird immer im Eigentum des Kopierenden sein, Zeitstempel wird angepasst

Hinweise mv:

- Benennt auch Verzeichnisse um bzw. verschiebt diese
- Verschieben ist in der Regel schnell, da kein physikalische Umkopieren, außer bei zwei physikalischen Laufwerken

Hinweise rmdir:

- Löschen funktioniert nur bei leeren Verzeichnissen
- In Benutzung befindliche Verzeichnisse werden nicht gelöscht

Versteckte Elemente:

- Dateien und Verzeichnisse mit vorangestelltem Punkt -> Versteckte Datei / Verzeichnis
- Anzeigen versteckter Dateien: $ls -a

4.6 Links

Innerhalb von Unix ist es möglich eine Datei zu verlinken (Windows: Verknüpfung). Damit kann eine Datei unter zwei verschiedenen Namen bekannt sein oder aber eine Datei ist in einem Ordner (über den Link) verfügbar, obwohl Sie in einem anderen Ordner liegt. Unix unterscheidet hierbei zwei Arten von Links.

4.6.1 Hard Link

Den Hardlink gibt es nur für Dateien und dies auch nur im gleichen Verzeichnissystem.

$ln <OrigDatei> <Linkname> legt link im gleichen Verzeichnis mit neuem Namen an
$ln <OrigDatei> <Verzeichnispfad> legt Link unter gleichen Name in anderem Verzeichnis an

Bei einem Hardlink kann die zugehörige Datei nur gelöscht werden, wenn alle Links gelöscht wurden. Die Anzahl der Links, die auf eine Datei verweisen kann mit ls -la eingesehen werden.

4.6.2 Symbolik Links

Symbolic Links können auch über Grenzen des Verzeichnissystems hinweg gelegt werden und sind auch für Verzeichnisse gültig.

$ ln -s <OrigDatei> <AlternativName>

Der symbolische Link wird vor allem an zwei Stellen eingesetzt. Einerseits lässt sich damit schnell eine Datei oder ein Ordner verschieben. Wenn man nicht weiß, ob ein Programm diese Datei braucht, dann kann man mit einem Symbolic link auf den neuen Ort verweisen. Auch bei der Installation einer speziellen Version einer Software in einem Verzeichnis kann man dem Verzeichnis einen Namen inkl. Versionsnummer geben. Wird ein spezieller Verzeichnispfad gefordert, um die Software zu finden, dann kann man mit einem symbolischen Link diese Verknüpfung herstellen:

```
1 #ln -s /opt/apache2-1-2 /opt/apache
```

Möchte man nun eine neuere Version installieren, die alten Dateien aber belassen, so installiert man die neue Version in einen Ordner mit der neuen Versionsnummer, löscht den Symbolic Link und legt einen neuen an.

4.7 Dateitypen

Unter Linux wird (mit Ausnahme von KDE und Gnome) der Dateityp nicht auf Grundlage der Dateiendung unterschieden. Dateien werden mit geeigneten Programmen geöffnet. Unter Unix werden oftmals Dateien ohne Endung gespeichert.

Mit file lässt sich herausfinden, um was für einen Dateityp es sich handelt.

```
$file <Dateiname>
```

```
1 $file /bin/ls
2 $file /etc/fstab
3 ...
4 ...
```

4.8 Besondere Dateien

Mit den Verzeichnissen und den Links kennen wir bereits zwei Arten besonderer Dateien. Bei Aufruf von ls -l kann man am ersten Buchstaben pro Zeile erkennen, ob es sich um ein Verzeichnis (d) oder einen symbolischen Link (l) handelt. Es gibt noch weitere besondere Dateien.

Es gibt so genannte special files. Dies sind Dateien, die einen Zugriff auf Peripheriegeräte ermöglichen. Man findet sie im Verzeichnis /dev. Sie tragen ein kleines c oder b als Kennzeichen. Man kann diese Direktzugriffe auf die Peripherie mit den normalen Dateizugriffen durchführen, sollte dies im Normalfall aber vermeiden, um die Dateien nicht durch Fehlzugriffe zu beschädigen. Im Allgemeinen gibt es Programme, die den Zugriff steuern. So wird der Zugriff auf den Drucker (/dev/lp0) üblicherweise über das Programm lp oder lpr geregelt. Das verhindert, dass sich verschiedene Benutzer gegenseitig die Ausdrucke durcheinander bringen.

Dateien mit einem s sind so genannte Sockets, und Dateien mit einem p sind named pipes. Beide dienen zur Kommunikation zwischen Programmen.

Mit ls -l sieht man auch welchen Typ von Datei es gibt

d - für Directory
l - symbolic Link
s - Sockets zur Interprogrammkommunikation
c oder b - device files (i.d.R im /dev -Verzeichnis)
p - named Pipes zur Interprogrammkommunikation

4.9 Einsicht in und Bearbeitung von Dateien

An dieser Stelle sein noch zwei kleine Programme vorgestellt, die unter Linux/Unix eine schnelle Bearbeitung von Dateien erlauben.

Mit less lässt sich schnell ein Blick in den Inhalt von (Text-)Dateien werfen. Da dieser nur lesend ist, bietet sich less an, um Dateien anzusehen, die man nicht unbedingt ändern will oder für die man sowieso kein Schreibrecht hat.

$ less <Pfad>/<Dateiname>

$ less /etc/fstab -> q für quit

Less ist dabei ein wichtiges Hilfsmittel um Bildschirmausgaben von Programmen durchzusehen, wenn diese länger als die Bildschirmseite sind. Dies erfolgt mit einer Umleitung der Ausgabe auf der Shell in den Viewer less:

$ls -la /etc | less

Texteditoren sind unter Linux/Unix eines der wichtigsten Arbeitsmittel. Mit ihnen kann man

- Konfigurationsdateien bearbeiten
- Quellcode schreiben
- Notizen anlegen
- Ergebnisse von Bildschirmausgaben umleiten und einesehen.
-...

Es gibt deshalb mit vi, ed, pico, nano, emacs, ... eine vielzahl von Editoren, die für Linux verfügbar sind. Ein kleiner handlicher Editor ist der nano (auf manchen Systemen auch als pico verfügbar).

nano <neue Datei> legt eine neue Datei an
nano <Dateiname> öffnet eine Datei

Die am Fußrand angezeigten Buchstaben zur Steuerung lassen sich mit der Strg-Taste aufrufen.

4.10 Weiter nützliche Programme

Neben den bereits vorgestellten Programmen gibt es noch viel weitere, die den Umgang mit Dateien erleichtern.

cat:

$cat <Dateiname>

```
1 $ cat /etc/passwd
2
```

touch:

Leere Datei anlegen

$ touch <Dateiname>

```
1
```

grep:

Nach einer bestimmten Zeichenfolge innerhalb einer Datei suchen

$ grep <Zeichenfolge> <Dateiname>

```
1 $ grep hallo beispieldatei.txt
```

find:

$ find <Pfad> -name <Dateiname>

```
1
```

4.11 Ausgaben umleiten

Programmen, die in der Shell gestartet werden, stehen drei Kanäle zur Verfügung:

- stdin: Standardeingabe 0 (norm. Tastatur)
- stdout: Standardausgabe 1 (norm. Terminal/Bildschirm)
- stderr: Standardfehler 2 (norm. Terminal/Bildschirm)

Beim arbeiten im Terminal bietet die Shell verschiedene Möglichkeiten, die Ausgabe einzelner Programme umzuleiten.

Umleitung der Ausgabe mit >:

stdout -> Datei:

```
1 $ ls -all /etc/ > textdatei
```

Umleitung der Eingabe mit <:

Datei -> stdin:

```
1 $ tr -d '0-9' < textdatei
```

Der Pipe-Operator |

Der Pipe-Operator (kurz für Pipeline) leitet die Ausgabe eines Befehls direkt an einen anderen Befehl weiter.

$befehl1 | befehl2 | ...

```
1 $ find | grep textdatei
2
3 $ ls /dev/ | grep tty
4
```