2.3 Lektion 2

Zertifikat:	Linux Essentials
Version:	1.6
Thema:	2 Sich auf einem Linux-System zurechtfinden
Objective:	2.3 Verzeichnisse verwenden und Dateien auflisten
Lektion:	2 von 2

Einführung

Das Unix Betriebssystem wurde Mitte der 1960er Jahre für Großrechner entwickelt. Viele **Benutzer** teilten sich solche Rechner und griffen **über *Terminals* auf** die **Ressourcen** des Systems zu. Diesem Ansatz folgen auch heutige Linux-Systeme. So sprechen wir immer noch von “Terminals” zur Eingabe von Befehlen in die Shell, und jedes Linux-System ist so organisiert, dass viele Benutzer auf einem einzigen System einfach anzulegen sind.

Heimatverzeichnisse

Dies ist ein Beispiel für ein normales Dateisystem unter Linux:

```
$ tree -L 1 /
/
├── bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── lib
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

Die meisten dieser Verzeichnisse sind auf **allen Linux-Systemen zu finden**: Von Servern über Supercomputer bis hin zu winzigen eingebetteten Systemen kann ein erfahrener Linux-Benutzer davon ausgehen, dass er den Befehl `ls` in `/bin` findet, dass er die Systemkonfiguration durch Anpassung von Dateien in `/etc` ändert und dass er Systemprotokolle in `/var` liest.

Die Positionen dieser Dateien und Verzeichnisse definiert der ***Filesystem Hierarchy Standard*** **(FHS**), den eine spätere Lektion behandelt. Sie werden im täglichen Umgang mit Linux mehr über die Inhalte dieser Verzeichnisse lernen, aber fürs erste sollten Sie sich merken:

- Änderungen, die Sie im Root-Dateisystem vornehmen, **betreffen alle Benutzer** und

- Änderungen von Dateien im Root-Dateisystem erfordern **Administratorrechte**.

Das bedeutet, dass es normalen Nutzern untersagt ist, diese Dateien zu ändern oder sogar zu lesen; wir werden das Thema Berechtigungen in einem späteren Abschnitt behandeln.

Nun konzentrieren wir uns auf das Verzeichnis `/home`, das an dieser Stelle bereits etwas vertraut sein sollte:

```
$ tree -L 1 /home
/home
├── user
├── michael
└── lara
```

Unser Beispielsystem hat drei normale Benutzer, und jeder unserer Benutzer hat seinen eigenen Bereich, in dem er Dateien und Verzeichnisse erstellen und ändern kann, ohne seine Nachbarn zu beeinträchtigen. In der vorherigen Lektion haben wir mit der folgenden Dateistruktur gearbeitet:

```
$ tree /home/user
user
└── Documents
    ├── Mission-Statement
    └── Reports
        └── report2018.txt
```

Tatsächlich sieht ein echtes Dateisystem aber eher so aus:

```
$ tree /home
/home
├── user
│   └── Documents
│       ├── Mission-Statement
│       └── Reports
│           └── report2018.txt
├── michael
│   ├── Documents
│   │   └── presentation-for-clients.odp
│   └── Music
```

…dasselbe gilt für `lara`.

Unter Linux ist `/home` ähnlich wie ein Mehrfamilienhaus: Viele Nutzer haben ihren eigenen, abgetrennten Bereich. Versorgung und Wartung des Gebäudes selbst liegt hingegen in der Verantwortung des Hausverwalters namens *root*.

Der besondere relative Pfad für das Heimatverzeichnis

Wenn Sie eine neue Terminalsitzung unter Linux starten, sehen Sie eine ähnliche Eingabeaufforderung:

```
user@hostname ~ $
```

Die Tilde (`~`) repräsentiert unser *Heimatverzeichnis*. Wenn Sie den Befehl `ls` ausführen, sehen Sie einige bekannte Ausgaben:

```
$ cd ~
$ ls
Documents
```

Was wir nun also über Linux wissen: Es ist einem Mehrfamilienhaus ähnlich, in dem viele Benutzer in `/home` wohnen. Der Bereich des Benutzers `user` wird sich folglich von dem des Benutzers `michael` unterscheiden. Das werden wir mit dem Befehl `su` ("switch user") zeigen, der den Benutzer wechselt.

```
user@hostname ~ $ pwd
/home/user
user@hostname ~ $ su - michael
Password:
michael@hostname ~ $ pwd
/home/michael
```

Die Bedeutung von `~` ändert sich je nach Benutzer: Für `michael` ist der absolute Pfad von `~` `/home/michael`, für `lara` ist er `/home/lara` und so weiter.

Relative-to-Home-Pfade

Die Verwendung von `~` in Befehlen ist sehr praktisch, vorausgesetzt, Sie wechseln nicht den Benutzer. Betrachten wir das folgende Beispiel für `user`, der eine neue Sitzung begonnen hat:

```
$ ls
Documents
$ cd Documents
$ ls
Mission-Statement
Reports
$ cd Reports
$ ls
report2018.txt
$ cd ~
$ ls
Documents
```

Beachten Sie, dass Benutzer eine neue Sitzung stets in ihrem Heimatverzeichnis beginnen. In diesem Beispiel ist `user` in das Unterverzeichnis `Documents/Reports` gereist. Mit dem Befehl `cd ~` ist er zum Ausgangspunkt zurückgekehrt; dasselbe hätte er mit dem Befehl `cd` ohne Argumente erreicht:

```
$ cd Documents/Reports
$ pwd
/home/user/Documents/Reports
$ cd
$ pwd
/home/user
```

Eine letzte Anmerkung: Wir können die Home-Verzeichnisse *anderer* Benutzer angeben, indem wir den Benutzernamen direkt nach der Tilde angeben, zum Beispiel:

```
$ ls ~michael
Documents
Music
```

Beachten Sie, dass das nur funktioniert, wenn `michael` uns die Erlaubnis erteilt hat, den Inhalt seines Home-Verzeichnisses zu sehen.

Nehmen wir an, `michael` möchte die Datei `report2018.txt` im Home-Verzeichnis von `user` lesen. Er hat die Erlaubnis dazu und kann den Befehl `less` verwenden.

```
$ less ~user/Documents/Reports/report2018.txt
```

Jeder Dateipfad, der das Zeichen `~` enthält, wird als *Relative-to-Home*-Pfad bezeichnet.

Versteckte Dateien und Verzeichnisse

In der vorangegangenen Lektion haben wir die Option `-a` für den Befehl `ls` und mit `ls -a` die beiden speziellen relativen Pfade `.` und `..` eingeführt. Die Option `-a` listet *alle* Dateien und Verzeichnisse auf, einschließlich *versteckter* Dateien und Verzeichnisse.

```
$ ls -a ~
.
..
.bash_history
.bash_logout
.bash-profile
.bashrc
Documents
```

Versteckte Dateien und Verzeichnisse beginnen immer mit einem **Punkt (`.`).** Standardmäßig enthält das Heimatverzeichnis eines Benutzers viele versteckte Dateien. Darin sind häufig benutzerspezifische Konfigurationseinstellungen abgelegt, und sie sollten nur von einem erfahrenen Benutzer geändert werden.

Die Long-list-Option

Der Befehl `ls` hat viele Optionen, um sein Verhalten zu ändern. Schauen wir uns eine der meistgenutzten an:

```
$ ls -l
-rw-r--r-- 1 user staff      3606 Jan 13  2017 report2018.txt
```

`-l` erstellt eine “long list” (“lange Liste”). Dateien und Verzeichnisse belegen in der Ausgabe jeweils eine Zeile, und es werden zusätzliche Informationen über jede Datei und jedes Verzeichnis angezeigt.

`-rw-r--r--`: Dateityp und Berechtigungen der Datei. Beachten Sie, dass eine normale Datei mit Bindestrich beginnt und ein Verzeichnis mit `d`.

`1`: Anzahl der Links zur Datei.

`user staff`: Gibt den Eigentümer der Datei an, in diesem Fall also `user`. Zudem ist die Datei der Gruppe `staff` zugeordnet.

`3606`: Größe der Datei in Bytes.

`Jan 13 2017`: Zeitstempel der letzten Änderung an der Datei.

`report2018.txt`: Name der Datei.

Themen wie Eigentümer, Berechtigungen und Links werden in späteren Lektionen behandelt. Wie Sie sehen, ist die -List-Variante von `ls` oft dem Standardaufruf vorzuziehen.

Weitere `ls`-Optionen

Nachfolgend finden Sie einige Varianten, wie der Befehl `ls` am häufigsten verwendet wird. Wie Sie sehen, lassen sich mehrere Optionen für die gewünschte Ausgabe kombinieren.

`ls -lh`: Die Kombination von “langer Liste” (“long list”) mit “menschenlesbaren” (“human readable”) Dateigrößen gibt uns nützliche Suffixe wie `M` für Megabyte oder `K` für Kilobyte.

`ls -d */`: Die Option `-d` listet Verzeichnisse auf, aber nicht deren Inhalt. Kombiniert mit `*/` zeigt sie nur Unterverzeichnisse und keine Dateien.

`ls -lt`: Kombiniert “lange Liste” mit der Option, nach “Zeitpunkt der letzten Änderung” zu sortieren. Dateien mit den letzten Änderungen stehen oben, Dateien mit den ältesten Änderungen unten, wobei die Reihenfolge auch umgekehrt werden kann.

`ls -lrt`: Kombiniert “lange Liste” mit “Zeitpunkt der letzten Änderung” und `-r` für “umgekehrte Sortierung” (“reverse order”), so dass Dateien mit den letzten Änderungen am Ende der Liste stehen. Neben der Sortierung nach “Zeitpunkt der letzten Änderung” sind auch “Zeitpunkt des letzten Zugriffs” oder nach “Zeitpunkt der letzten Statusänderung” möglich.

`ls -lX`: Kombiniert “lange Liste” mit der Sortierung nach Dateiendungen (“eXtension”), um z.B. alle Dateien, die mit `.txt` oder mit `.jpg` enden, zusammenzufassen.

`ls -S`: sortiert nach Dateigröße, so wie `-t` nach Zeit oder `-X` nach Dateiendung, wobei die größten Dateien an erster Stelle stehen und die kleinsten zuletzt. Inhalte von Unterverzeichnissen sind übrigens von der Sortierung ausgenommen.

`ls -R`: Die Option `-R` bewirkt für den Befehl `ls`, dass er eine *rekursive* Liste ausgibt. Was bedeutet das?

Rekursion in Bash

Rekursion bezeichnet eine Situation, in der “etwas in sich selbst definiert” ist. Rekursion ist ein sehr wichtiger Begriff in der Informatik, aber hier ist seine Bedeutung viel einfacher. Betrachten wir unser Beispiel von vorhin:

```
$ ls ~
Documents
```

Wir wissen bereits, dass `user` ein Home-Verzeichnis hat und dass es in diesem Verzeichnis ein Unterverzeichnis gibt. `ls` hat uns bisher nur die Dateien und Unterverzeichnisse eines Ortes ausgegeben, kann uns aber den Inhalt dieser Unterverzeichnisse nicht anzeigen. In diesen Lektionen haben wir den Befehl `tree` verwendet, um den Inhalt vieler Verzeichnisse anzuzeigen. `tree` ist leider kein Standardwerkzeug von Linux und steht daher nicht immer zur Verfügung. Vergleichen Sie die Ausgabe von `tree` mit der von `ls -R` in den folgenden Beispielen:

```
$ tree /home/user
user
└── Documents
    ├── Mission-Statement
    └── Reports
        └── report2018.txt

$ ls -R ~
/home/user/:
Documents

/home/user/Documents:
Mission-Statement
Reports

/home/user/Documents/Reports:
report2018.txt
```

Wie Sie sehen, erhalten wir mit der Rekursiv-Option eine viel längere Liste von Dateien. Tatsächlich ist es so, als würden wir den Befehl `ls` im Heimatverzeichnis von `user` aufrufen und auf ein Unterverzeichnis treffen; daraufhin gehen wir in dieses Unterverzeichnis und führen den Befehl `ls` dort erneut aus.

Wir sehen die Datei `Mission-Statement` und ein weiteres Unterverzeichnis namens `Reports`. Wir gehen in das Unterverzeichnis und haben den Befehl `ls` erneut ausgeführt. Im Grunde sagt der Befehl `ls -R` der Bash: “Führe `ls` hier aus und wiederhole den Befehl in jedem Unterverzeichnis, das Du findest.”

Rekursion ist besonders wichtig bei Änderungen von Dateien wie dem Kopieren oder Entfernen von Verzeichnissen. Wenn Sie etwa das Unterverzeichnis `Documents` kopieren möchten, müssen Sie eine rekursive Kopie vornehmen, um den Befehl auf alle Unterverzeichnisse auszuweiten.

Geführte Übungen

1. Nutzen Sie die folgende Dateistruktur, um die folgenden drei Fragen zu beantworten:

```
/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

- Welcher Befehl wechselt ins Verzeichnis `network` unabhängig vom aktuellen Standort?
- Welchen Befehl kann `user` eingeben, um von `/etc/udev` ins Verzeichnis `Documents` zu wechseln? Geben Sie den kürzestmöglichen Pfad an.
- Welchen Befehl kann `user` eingeben, um in das Verzeichnis `music` des Benutzers `michael` zu wechseln? Nutzen Sie den kürzestmöglichen Pfad.

2. Betrachten Sie die folgende Ausgabe von `ls -lh`, um die nächsten beiden Fragen zu beantworten. Beachten Sie, dass Verzeichnisse mit einem `d` am Zeilenanfang gekennzeichnet sind.

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0067.jpg
-rwxrwxrwx  1 eric eric  1.6M Jul 18  2011 img_0074.jpg
-rwxrwxrwx  1 eric eric  1.8M Jul 18  2011 img_0075.jpg
-rwxrwxrwx  1 eric eric   46K Jul 18  2011 scary.jpg
-rwxrwxrwx  1 eric eric  469K Jan 29  2018 Screenshot from 2017-08-13
21-22-24.png
-rwxrwxrwx  1 eric eric  498K Jan 29  2018 Screenshot from 2017-08-14
21-18-07.png
-rwxrwxrwx  1 eric eric  211K Jan 29  2018 Screenshot from 2018-01-06
23-29-30.png
-rwxrwxrwx  1 eric eric  150K Jul 18  2011 tobermory.jpg
drwxrwxrwx  6 eric eric  4.0K Apr 26  2011 Tokyo/
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 081.jpg
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 085.jpg
-rwxrwxrwx  1 eric eric  944K Jul 18  2011 Toronto 152.jpg
-rwxrwxrwx  1 eric eric  728K Jul 18  2011 Toronto 173.jpg
drwxrwxrwx  2 eric eric  4.0K Jun  5  2016 Wallpapers/
```

   - Welche Datei steht zu Beginn, wenn Sie den Befehl `ls -lrS` ausführen?
   - Bitte beschreiben Sie, welche Ausgabe Sie von dem Befehl `ls -ad */` erwarten.

Offene Übungen

1. Führen Sie den Befehl `ls -lh` in einem Verzeichnis aus, das Unterverzeichnisse enthält. Beachten Sie die angezeigte Größe dieser Verzeichnisse. Scheint Ihnen die Dateigröße korrekt? Entspricht sie dem Inhalt aller Dateien in diesem Verzeichnis?
2. Hier ein neuer Befehl zum Ausprobieren: `du -h`. Führen Sie diesen Befehl aus und beschreiben Sie dessen Ausgabe.
3. Auf vielen Linux-Systemen können Sie `ll` eingeben und erhalten die gleiche Ausgabe wie bei `ls -l`. Beachten Sie jedoch, dass `ll` *kein* Befehl ist. `man ll` wird beispielsweise darauf hinweisen, dass keine entsprechende Manpage existiert. Es ist ein Beispiel für einen *Alias*. Inwiefern können Aliase nützlich sein?

Zusammenfassung

In dieser Lektion haben Sie gelernt, dass

- jeder Linux-Benutzer ein Heimatverzeichnis hat,
- das Heimatverzeichnis des aktuellen Benutzers über `~` zu erreichen ist,
- jeder Dateipfad, der `~` verwendet, als *Relative-to-home-Pfad* bezeichnet wird.

Sie haben zudem einige der meistgenutzten Optionen für `ls` kennengelernt:

`-a` (all)
: Gibt alle Dateien/Verzeichnisse aus, einschließlich der versteckten.

`-d` (directories)
: Gibt alle Verzeichnisse aus, nicht deren Inhalt.

`-h` (human readable)
: Gibt Dateigrößen in einem menschenlesbaren Format aus.

`-l` (long list)
: Liefert zusätzliche Details mit einer Datei/einem Verzeichnis pro Zeile.

`-r` (reverse)
: Kehrt die Reihenfolge einer Sortierung um.

`-R` (recursive)
: Listet jede Datei, einschließlich der Dateien in allen Unterverzeichnissen.

`-S` (size)
: Sortiert nach Dateigröße.

`-t` (time)
: Sortiert nach dem Zeitpunkt der letzten Änderung.

`X` (eXtension)
: Sortiert nach Dateiendung.