

| Linux Grundlagen

- [Linux Einstieg](#)
- [Praktische Übung](#)

Linux Einstieg

Linux ist ein freies und quelloffenes Betriebssystem, das für die Verwendung auf Computern, Servern, mobilen Geräten und anderen Geräten konzipiert ist. Es wurde ursprünglich von Linus Torvalds entwickelt und wird seitdem von einer weltweiten Gemeinschaft von Programmierern ständig weiterentwickelt. Linux ist ein zuverlässiges, sicheres und robustes Betriebssystem und wird wegen seiner Flexibilität und Kosteneffizienz häufig im professionellen Bereich eingesetzt. Es wird auch häufig als Alternative zu proprietären Betriebssystemen wie Windows verwendet.

Das zentrale Kernstück des Betriebssystems ist der Linux-Kernel. Er ist verantwortlich für das Verwalten der Ressourcen des Computers, das Ausführen von Programmen, das Verarbeiten von Anweisungen und vieles mehr. Es ist das Herzstück des Betriebssystems, das alle anderen Komponenten zusammenführt.

Wie unter Windows auch, können unter Linux Anwenderprogramme ausgeführt werden. Ausführbare Windows-Programme und Linux-Programme sind hierbei in erster Linie nicht miteinander kompatibel, da die zugrundeliegende Architektur unterschiedlich ist. Viele Programme werden jedoch zwischenzeitlich auch speziell für Linux bereitgestellt.

Während bei Windows viele Programme aus dem Internet heruntergeladen und installiert werden, hat sich unter Linux die Software-Verwaltung über sogenannte Paketmanager etabliert. Paketmanager wie **apt** und **yum** ermöglichen es den Benutzern, Programme aus einer Vielzahl von Quellen zu installieren und zu verwalten. Die Installation kann dabei direkt über das Terminal vorgenommen werden. Die Programme müssen somit nicht manuell heruntergeladen werden um diese zu installieren oder aktualisieren.

Die Shell

Ein ganz spezielles Anwenderprogramm ist die **Shell** - die "Benutzeroberfläche". Es existieren viele verschiedene Shells - in diesem Modul werden wir mit der Bash (**B**ourne **a**gain **s**hell) arbeiten. Diese ist die Standardshell auf Linuxsystemen. Alle Shells stellen dem Benutzer eine Kommandozeile zur Verfügung, mit der Befehle eingegeben werden können, die direkt als Systemaufrufe an das Betriebssystem weitergeleitet werden.

image.png or type unknown

Es gibt auch grafische Shells für Linux. Dies sind wie bereits von Windows bekannt, Desktop-Umgebungen welche mit der Maus bedient werden können. Bekannte Vertreter sind hierbei Gnome, Unity, KDE, oder Cinnamon.

Die grafischen Shells spielen in diesem Modul keine Rolle. Da Server oftmals ohne Bildausgabe ausgestattet sind, wird auf Server-Umgebungen bevorzugt die Text-Konsole verwendet.

Dateisystem

Wie Windows arbeitet Linux ebenfalls mit Dateien. Im Unterschied zu Windows, hat aber nicht jedes Laufwerk einen eigenen Dateibaum, sondern es existiert ein einziger Dateibaum wo auch die Laufwerke mit eingebunden werden.

image.png or type unknown

Das Root-Verzeichnis /

Das Root-Verzeichnis ist das oberste Verzeichnis im Linux Dateibaum. Es ist eine Partition, in der das Root-Verzeichnis eingehängt ist. Weiter Laufwerke und Partition werden an beliebigen Stellen im Dateibaum eingehängt (mounted).

Im Wurzelverzeichnis / darf sich neben den weiteren Verzeichnissen /bin /etc /home /opt /tmp /usr und /var nur eine einzige normale Datei befinden. Dies ist der Linux-Kernel. Heutzutage ist der Linux-Kernel jedoch meistens im Verzeichnis /boot ausgelagert.

/bin

In diesem Verzeichnis liegen sogenannte Binaries, also binäre Programme, jedoch nur jene, die für die wichtigsten Arbeiten am System benötigt werden. Alle User haben in diesem Verzeichnis Lesezugriff.

/boot

Hier liegt der Linux-Kernel und alle Dateien, die er zum Starten benötigt. Nur der root-Benutzer hat auf dieses Verzeichnis Zugriff.

/dev

Hier befinden sich die Gerätedateien. Dies sind Schnittstellen zum Kernel, die ein bestimmtes Gerät bezeichnen. Unter Linux kann fast jede Hardware mit einer solchen Gerätedatei angesprochen werden. Wichtig ist, dass diese Dateien keinen physischen Platz auf der Festplatte beanspruchen. Sie existieren nur virtuell.

/etc

Alle systemweit gültigen Konfigurationsdateien (Netzwerk, Bootmanager, Systemstartskripte, u.s.w) werden hier gespeichert. Auch die Dateien für die Benutzerverwaltung (z.B. "passwd" und "shadow" mit den Benutzern und deren Passwörtern) sind hier abgelegt.

Viele Prozesse müssen in der Lage sein, diese Dateien zu lesen, um bestimmte Informationen zu erhalten. Daher muss das Verzeichnis für alle Benutzer lesbar sein.

/home

Jeder dem Betriebssystem bekannte Benutzer hat sein eigenes Verzeichnis, welches in der Regel denselben Namen wie der Benutzer trägt. Wir nennen solche Verzeichnisse die "Heimatverzeichnisse" der Benutzer. Sie befinden sich alle (mit Ausnahme des "root"-Verzeichnisses des Systemadministrators) im Verzeichnis /home. Nach dem Einloggen (Anmeldung am System mit Benutzernamen und Passwort) befinden sich alle Benutzer immer in ihrem Home-Verzeichnis.

In der Regel darf ein Benutzer in seinem eigenen Home-Verzeichnis alles tun - auch Dateien anlegen, löschen und ändern. Ausserhalb seines Verzeichnisses hat ein normaler Benutzer in der Regel nur Leserechte - er darf dort keine Dateien ändern, anlegen oder löschen.

/root

Dies ist das Heimatverzeichnis des Systemadministrators "root". Aber warum befindet sich dieses Verzeichnis nicht im Verzeichnis /home? Das ist ganz einfach: /home wird oft als eigene Partition eingerichtet. Diese Partition wird aber nur beim Systemstart eingebunden. Sollte ein Fehler auftreten, muss der Systemadministrator in der Lage sein, das System zu starten, ohne alle Partitionen einzubinden. Daher muss auf dem Root-Verzeichnis bereits ein funktionierendes System vorhanden sein. Das Home-Verzeichnis ist jedoch ein wesentlicher Bestandteil eines funktionierenden Systems. Deshalb befindet sich das Home-Verzeichnis von "root" direkt auf dem Root-Verzeichnis und in der gleichen Partition.

In frühen Unix-Systemen war das Home-Verzeichnis von "root" tatsächlich das Root-Verzeichnis, aber da ein Home-Verzeichnis immer viele Dateien enthält, wurde dies zu unübersichtlich und sein Home-Verzeichnis wurde in das /root-Verzeichnis verschoben.

/sbin

Wie in /bin gibt es auch hier Binärdateien, allerdings solche, die hauptsächlich vom Systemadministrator verwendet werden - sozusagen "Systembinärdateien".

/tmp

In diesem Verzeichnis werden, wie der Name schon sagt, temporäre Dateien gespeichert, die nach der Benutzung nicht mehr benötigt werden. Alle Benutzer müssen hier schreiben können, aber es ist kein guter Ort, um die eigenen Daten sicher aufzubewahren. Der gesamte Inhalt wird normalerweise bei einem Neustart des Systems gelöscht.

/usr

Dieses Verzeichnis enthält alle wichtigen Programme, die das System anbietet. Dabei steht "usr" nicht für "user", wie oft fälschlicherweise angenommen wird, sondern für "Unix System Resources".

Analog zum Stammverzeichnis finden wir die Unterverzeichnisse **/usr/bin**, **/usr/lib** und **/usr/sbin**, deren Bedeutung der des Stammverzeichnisses entspricht.

- **/usr/local** dient der Trennung von distributionseigenen und fremden, zusätzlich installierten Programmen. Unter /usr/local finden wir die gleichen Verzeichnisse wie im Verzeichnis /usr selbst. Selbstkompilierte Programme sollten immer in /usr/local und nie direkt in /usr installiert werden und die Konfigurationsskripte der Programme sehen dies in der Regel als Standard vor.

- **/usr/share** enthält architekturunabhängige Daten, die verschiedene Programme zum Betrieb benötigen und die nicht verändert werden.
- **/usr/src** enthält den Quellcode für alle Programme des Standardsystems.

/var

Dieses Verzeichnis enthält Daten, die oft von Programmen neu geschrieben werden müssen. Ursprünglich waren diese Daten auch unter /usr zu finden, aber der Wunsch, /usr schreibgeschützt mounten zu können, machte eine Trennung notwendig. Hier werden Mails auf einem Mailserver zwischengespeichert (/var/spool) und alle Programme schreiben hier ihre Log-Dateien (/var/log).

Daneben gibt es noch einige andere Verzeichnisse und Unterverzeichnisse, die zum Teil leer sind oder nur symbolische Links auf andere Verzeichnisse enthalten und oft aus Kompatibilitätsgründen existieren.

/opt

Nachdem immer mehr grosse Programmpakete installiert wurden, kam die Überlegung auf, diese nicht mehr in das Verzeichnis /usr zu legen, sondern ein eigenes Verzeichnis dafür anzulegen. Tatsächlich nutzen aber nur wenige Programme diese Möglichkeit von sich aus und auf vielen Systemen bleibt dieses Verzeichnis leer.

Praktische Übung

1. Erstellen Sie eine neue EC2 Instanz. Klicken Sie anschliessend auf den Button **Mit Instance verbinden** um eine Web-Bash zu öffnen:

 or type unknown

2. Selektieren Sie den Tab **EC2-Instance-Verbindung** und klicken Sie auf den Button **Verbinden**

Sie sehen nun folgende Ausgabe nach dem erfolgreichen Login.

```
┌ ┌ ┌ )
└ ( / Amazon Linux 2 AMI
└ \ └ └

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.
```

Die Anmeldung ist unter dem Benutzer **ec2-user** erfolgt. Sie können dies verifizieren, indem Sie nun den folgenden Befehl eintippen und mit Enter bestätigen:

```
whoami
```

Frage: Was für eine Ausgabe sehen Sie und was bewirkt der Befehl whoami vermutlich?

Der Benutzer ec2-user hat nur eingeschränkte Rechte. Mit diesem Benutzer können Sie beispielsweise keine System-Dateien einsehen oder verändern. Aus diesem Grund müssen Sie nun dem Betriebssystem

mitteilen, dass Sie gerne nun mit dem Administrator-Benutzer arbeiten möchten. Verwenden Sie hierzu den Befehl **sudo su** und bestätigen Sie mit Enter.

```
[ec2-user@ip-172-31-95-132 ~]$ sudo su
[root@ip-172-31-95-132 ec2-user] #
```

Frage: Was hat sich nun an der Text-Eingabe verändert. Beachten Sie den obenstehenden Output genau. Wie lautet der Name des Admin-Benutzers?

Mit dem Admin-Benutzer können Sie uneingeschränkt Dateien verändern auf dem Linux-System. Man spricht hier von dem sogenannten Root-Zugriff. Damit können Sie aber auch vieles kaputt machen. Setzen Sie daher auf einer produktiven Umgebung Befehle mit Bedacht ab.

Navigation im Dateisystem

Sie sind es sich gewohnt, dass Ordner visuell dargestellt werden um darin zu navigieren. In der Shell ist dies etwas anders. Die Ordner und Dateien werden als Text aufgelistet. Als erstes möchten wir nun herausfinden, in welchem Verzeichnis wir uns genau befinden. Dazu können Sie den Befehl **pwd** verwenden:

```
[root@ip-172-31-95-132 ec2-user] # pwd
/home/ec2-user
```

Vermutlich befinden sie sich ebenfalls im Home Verzeichnis des Benutzers ec2-user. Das Home-Verzeichnis entspricht hierbei dem C:/Users/<Benutzer> Verzeichnis unter Windows. Sie finden hier die persönlichen Daten des Benutzers.

Wenn Sie nun sehen möchten, ob sich im Verzeichnis des Benutzers ec2-user irgendetwas Dateien oder Verzeichnisse befinden, können Sie den Befehl **ls** absetzen. ls steht für Listing und listet alle Elemente in einem Verzeichnis auf:

```
[root@ip-172-31-95-132 ec2-user] # ls
[root@ip-172-31-95-132 ec2-user] #
```

Sie sehen nun vermutlich kein Resultat. Es hat somit vermutlich keine Dateien. Dies stimmt jedoch nicht ganz, denn genau wie unter Windows, kann es versteckte Dateien geben. Diese können Sie aufrufen, indem Sie dem Befehl **ls** einen Parameter mitgeben (auch als Flag bekannt). Wir geben dem Befehl **ls** nun den Parameter **a** für all mit. Damit listen wir alle Dateien und Verzeichnisse auf, auch die versteckten. Ein Parameter wird mit einem Abstand mit einem Bindestrich dem gewünschten Befehl

angehängt:

```
[root@ip-172-31-95-132 ec2-user]# ls -a
.  ..  .bash_logout  .bash_profile  .bashrc  .ssh
```

Nun werden die versteckten Dateien und Verzeichnisse ersichtlich. Sie können sich nun vorstellen, dass dies bei vielen Dateien und Verzeichnissen eher unübersichtlich ist. Aus diesem Grund existiert ein weiterer Parameter. Mit dem Parameter **l** (kleines L) kann eine Listendarstellung erzwungen werden. Die beiden Parameter **l** und **a** können dabei kombiniert werden. Dies sieht so aus:

```
[root@ip-172-31-95-132 ec2-user]# ls -la
total 12
drwx----- 3 ec2-user ec2-user 74 Oct 17 22:30 .
drwxr-xr-x 3 root      root      22 Oct 17 22:30 ..
-rw-r--r-- 1 ec2-user ec2-user 18 Jul 15  2020 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user 193 Jul 15  2020 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user 231 Jul 15  2020 .bashrc
drwx----- 2 ec2-user ec2-user 29 Oct 17 22:30 .ssh
```

Nun sind viele Informationen ersichtlich.

- Die erste Spalte gibt die Berechtigungen der Dateien und Verzeichnissen aus. Sie erinnern sich aus dem Modul 117. (R = Read, W = Write und X = Execute). Der erste Buchstabe gibt hierbei an, ob es sich um eine Datei oder ein Verzeichnis handelt (d = Directory (Verzeichnis) / - = Datei)
- Die dritte Spalte gibt den Besitzer der Datei aus. Wir sehen, die meisten Elemente gehören dem Benutzer ec2-user und ein Element dem Benutzer Root
- Die vierte Spalte gibt die Gruppe welcher die Datei zugeordnet ist.
- Die fünfte Spalte gibt die Grösse der Datei in Bytes an.
- Die sechste bis achte Spalte gibt das Datum und die Uhrzeit an, wo die Datei das letzte Mal verändert wurde.
- In der letzten Spalte ist der Name der Datei oder des Verzeichnisses ersichtlich.

Weiter sehen Sie im Verzeichnis zwei komische Elemente bestehend aus einem Punkt (.) und einem Doppelpunkt (..). Diese repräsentieren das aktuelle Verzeichnis (.) sowie das übergeordnete Verzeichnis (..). Wir benötigen diese Elemente zur Navigation innerhalb des Dateisystems.

Nun können Sie den Inhalt in einem Verzeichnis anzeigen lassen und interpretieren.

Navigieren

Sie haben nun gelernt, wie Sie den Inhalt in einem Verzeichnis anzeigen können. Nun wird es Zeit, dass Sie sich auch andere Verzeichnisse anschauen können. Aus diesem Grund werden Sie nun als erstes ein Verzeichnis höher Navigieren. Dazu verwenden Sie den Befehl **cd**. Dies ist die Abkürzung für change directory (Ändere das Verzeichnis). Der Befehl **cd** erfordert immer die Angabe des Zielverzeichnisses. Wenn wir nun eine Verzeichnis-Ebene nach oben navigieren möchten, setzen Sie folgenden Befehl **cd ..** ab:

```
[root@ip-172-31-95-132 ec2-user]# cd ..  
[root@ip-172-31-95-132 home]#
```

Frage: Wie erkennen Sie den Namen des übergeordneten Verzeichnisses nachdem Sie gewechselt haben, ohne dass Sie den Befehl **pwd** verwenden? Wie lautete der Name des Verzeichnisses?

Kontrollieren Sie anschliessend mit dem Befehl **pwd** ob Ihre Vermutung korrekt ist.

Navigieren Sie nun wieder in das Verzeichnis des Benutzers **ec2-user**:

```
[root@ip-172-31-95-132 ec2-user]# cd ..  
[root@ip-172-31-95-132 home]#
```

Geheimtipp: Wenn Sie im Dateisystem navigieren, können Sie die Tab-Taste verwenden. Es handelt sich hierbei um eine vervollständigungs-Funktion. Tippen Sie **cd** und anschliessend die Tab-Taste. Sie sehen nun alle verfügbaren Verzeichnisse. Schreiben Sie nun den ersten Buchstaben des gewünschten Verzeichnisses und tippen Sie erneut Tab. Der Name wird nun im Befehl automatisch ergänzt.

Sie können nun im Dateisystem navigieren.

Dateien schreiben

Es gibt verschiedene Möglichkeiten wie Sie unter Linux eine Datei erstellen oder bearbeiten können. Hierzu sind vorallem zwei Text-Editoren bekannt, Vim und Nano. Einen Text-Editor benötigen Sie, sobald z. B. eine Konfigurationsdatei angepasst werden soll.

In dieser Übung werden wir den Text-Editor Nano verwenden, da dieser etwas einfacher ist. Falls Sie mit Vim bereits vertraut sind können die Aufgaben natürlich auch mit Vim gelöst werden.

Prüfen Sie als erstes mit den Befehl **pwd** ob Sie sich im korrekten Verzeichnis /home/ec2-user befinden. Der Befehl **pwd** gibt den Pfad aus, in dem sich der Benutzer aktuell befindet (Present Working Directory / Aktuelles Arbeitsverzeichnis). Erstellen Sie nun eine neue Datei mit dem Namen Testdatei.txt indem Sie nano aufrufen und als erstes Argument den Namen der Datei übergeben:

```
nano Testdatei.txt
```

Nano öffnet nun im Editiermodus und Sie können Ihren Inhalt schreiben. Um die Datei zu speichern und beenden verwenden Sie die Tastenkombination CTRL + X. Nano wird Sie nun fragen, ob der gesetzte Dateinamen in Ordnung ist. Bestätigen Sie hier einfach mit Enter.

Sie können nun den Inhalt der Datei anzeigen. Eine Möglichkeit ist natürlich, dass Sie die Datei mit Nano wieder im Bearbeitungsmodus öffnen. Sie können aber auch mit dem Befehl **cat** den Inhalt der Datei direkt im Terminal darstellen lassen. Als Argument geben Sie den Dateinamen der gewünschten Datei an.

```
[root@ip-172-31-95-132 ec2-user]# cat Testdatei.txt
Das ist der Inhalt meiner Testdatei:

1. Listenelement
2. Listenelement
3. Listenelement
[root@ip-172-31-95-132 ec2-user]#
```

Dateien verschieben

Falls Sie mit dem Namen der Datei nicht einverstanden sind können Sie diesen einfach umbenennen. Hierzu existiert der Befehl mv. Der Name leitet sich ab von move, denn der Befehl kann auch Dateien von einem Ort zu einem anderen Ort verschieben.

Als erstes werden wir aber anschauen, wie eine Datei mit mv umbenannt werden kann:

```
[root@ip-172-31-95-132 ec2-user]# mv Testdatei.txt Testdatei-umbenannt.txt
```

Frage: Mit welchem Befehl können Sie nun kontrollieren ob die Umbenennung nun geklappt hat?

Nun werden wir die Datei aus dem Verzeichnis /home/ec2-user in das Verzeichnis für temporäre Dateien /tmp verschieben.

```
[root@ip-172-31-95-132 ec2-user]# mv Textdatei-umbenannt.txt /tmp/
```

Beachten Sie, dass dieser Befehl in dieser Form nur funktioniert, wenn Sie sich im selben Verzeichnis wie die zu verschiebende Datei befinden. Sollten Sie aktuell in einem anderen Verzeichnis sein (prüfen mit **pwd**) müssen Sie den vollen Pfad der Datei angeben:

```
[root@ip-172-31-95-132 ec2-user]# mv /home/ec2-user/Textdatei-umbenannt.txt /tmp/
```

Kontrollieren Sie nun ob die Datei im Verzeichnis /home/ec2-user ordnungsgemäss verschwunden ist. Wechseln Sie anschliessend in das Verzeichnis /tmp und prüfen Sie, ob Sie hier die Datei finden.

Dateien löschen

Wenn Sie eine Datei löschen möchten, können Sie dies mit dem Befehl **rm** bewerkstelligen. Der name **rm** leitet sich von remove ab und bedeutet sprichwörtlich entfernen.

Der Befehl **rm** hat keine Papierkorb-Funktion. Wird eine Datei mit **rm** gelöscht, dann ist diese auch wirklich weg.

Den Namen der zu löschenden Datei geben Sie als Argument dem Befehl **rm** mit:

```
[root@ip-172-31-95-132 tmp]# rm Textdatei-umbenannt.txt  
rm: remove regular file 'Textdatei-umbenannt.txt'?
```

Bestätigen Sie indem Sie **y** tippen.

Cheatsheet

Nachfolgend finden Sie die wichtigsten Linux-Befehle, welche Sie für dieses Modul benötigen werden:

Befehl	Bedeutung	Beispiel
ls	Listete den Inhalt in einem Verzeichnis auf. Mit dem Parameter -la werden auch versteckte Dateien angezeigt und das Ergebnis in Listenform dargestellt.	ls -la
cd	Wechselt in ein anderes Verzeichnis	cd /home/ec2-user cd ..
pwd	Zeigt den Pfad des aktuellen Arbeitsverzeichnisses	pwd
nano	Ein Editor mit der Textdateien erstellt und bearbeitet werden können. CTRL + X um eine Datei zu speichern	nano datei.txt
mv	Kann eine Datei verschieben oder umbenennen	mv alter-name.txt neuer-name.txt mv /home/ec2-user/datei.txt /tmp
rm	Löscht eine Datei	rm datei.txt