3.2 Lektion 1

Zertifikat:	Linux Essentials
Version:	1.6
Thema:	3 Die Macht der Befehlszeile
Lernziel:	3.2 Daten in Dateien suchen und extrahieren
Lektion:	1 von 2

Einführung

In dieser Lektion konzentrieren wir uns **auf die Umleitung oder Übergabe von Informationen** von einer Quelle zur anderen mit Hilfe spezifischer Werkzeuge. Die Linux-Befehlszeile leitet Informationen über bestimmte Standardkanäle um. Als **Standardeingabe** (*Standard Input*, ***stdin* oder Kanal 0**) **eines Befehls gilt die Tastatur**, und als **Standardausgabe** (*Standard Output*, ***stdout* oder Kanal 1**) der Bildschirm. Es gibt einen weiteren Kanal, der dazu dient, die **Fehlerausgabe** (*Standard Error*, ***stderr* oder Kanal 2**) eines Befehls oder die Fehlermeldungen eines Programms umzuleiten. Der Ein- und/oder Ausgabe kann umgeleitet werden.

Wenn wir einen Befehl ausführen, wollen wir manchmal bestimmte Informationen an den Befehl senden oder die Ausgabe in eine bestimmte Datei umleiten. Diese Funktionalitäten behandeln die nächsten beiden Abschnitte.

I/O-Umleitung

Die I/O-Umleitung ermöglicht es dem Benutzer**, Informationen** von oder zu einem Befehl mithilfe einer Textdatei **umzuleiten**. Wie bereits beschrieben, kann die **Standardeingabe, -ausgabe und -fehlerausgabe umgeleitet** werden, und die Informationen können aus Textdateien übernommen werden.

Umleitung der Standardausgabe

Um die **Standardausgabe in eine Datei umzuleiten**, nutzen wir den **Operator** `>`, gefolgt vom Namen der Datei. Wenn die **Datei** nicht existiert, wird eine **neue erstellt, andernfalls überschreiben** die Informationen die bestehende Datei.

Um den Inhalt der Datei, die wir gerade erstellt haben, zu sehen, können wir den Befehl `cat` verwenden, der standardmäßig **den Inhalt einer Datei auf dem Bildschirm** anzeigt. Auf der Man Page erfahren Sie mehr über dessen Funktionalitäten.

Das folgende Beispiel zeigt die Funktionalität des Operators. Zunächst wird eine neue Datei mit dem Text "`Hello World!`" erstellt.:

```
$ echo "Hello World!" > text
$ cat text
Hello World!
```

Beim zweiten Aufruf wird die gleiche Datei mit dem neuen Text überschrieben:

```
$ echo "Hello!" > text
```

```
$ cat text
Hello!
```

Um **neue Informationen am Ende der Datei hinzuzufügen**, nutzen wir den **Operator** `>>`, der auch eine neue Datei erstellt, wenn er keine existierende findet.

Das erste Beispiel zeigt das Hinzufügen des Textes. Wie man sieht, wurde der neue Text in der folgenden Zeile hinzugefügt:

```
$ echo "Hello to you too!" >> text
$ cat text
Hello!
Hello to you too!
```

Das zweite Beispiel zeigt, dass eine neue Datei erstellt wird:

```
$ echo "Hello to you too!" >> text2
$ cat text2
Hello to you too!
```

Umleitung der Standardfehlerausgabe (Standard Error)

Um nur die **Fehlermeldungen** umzuleiten, nutzt ein User den **Operator** `2>`**,** gefolgt vom Namen der Datei, in die die Fehler geschrieben werden sollen. Wenn die Datei nicht existiert, wird eine neue erstellt, andernfalls wird die Datei überschrieben.

Wie bereits erläutert, ist der Kanal für die Umleitung der **Standardfehlerausgabe *Kanal 2***. Bei der Umleitung der Standardfehlerausgabe muss der Kanal angegeben werden, im Gegensatz zu den anderen Standardausgaben, bei denen ***Kanal 1* standardmäßig gesetzt** ist. Der folgende Befehl sucht beispielsweise nach einer Datei oder einem Verzeichnis namens games und **schreibt nur den Fehler in die Datei** `text-error`, während er die Standardausgabe auf dem Bildschirm anzeigt:

```
$ find /usr games 2> text-error
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
$ cat text-error
find: `games': No such file or directory
```

Note Weitere Informationen über den Befehl `find` finden Sie in der Man Page.

Der folgende Befehl wird ohne Fehler ausgeführt, daher werden keine Informationen in die Datei `text-error` geschrieben:

```
$ sort /etc/passwd 2> text-error
$ cat text-error
```

Wie die Standardausgabe kann auch die Standardfehlerausgabe an eine Datei mit dem **Operator** `2>>` **angehängt werden**. Der neue Fehler wird am Ende der Datei hinzugefügt. Wenn die Datei nicht existiert, wird eine neue erstellt. Das erste Beispiel zeigt das Hinzufügen der neuen Informationen in die Datei, während das zweite Beispiel zeigt, dass der Befehl eine neue Datei erstellt, falls keine Datei dieses Namens gefunden wird:

```
$ sort /etc 2>> text-error
$ cat text-error
sort: read failed: /etc: Is a directory
$ sort /etc/shadow 2>> text-error2
$ cat text-error2
sort: open failed: /etc/shadow: Permission denied
```

Bei dieser Art der Umleitung werden **nur Fehlermeldungen in die Datei umgeleitet**, die **normale Ausgabe erscheint auf dem Bildschirm** bzw. geht an die Standardausgabe (*stdout*).

Es gibt eine **bestimmte Datei**, die technisch gesehen ein "`Daten-Mülleimer`", auch ***bit bucket* genannt**, ist, das heißt sie akzeptiert Eingaben, macht aber nichts damit: `/dev/null`. Sie können alle unwichtigen Informationen, die Sie nicht anzeigen oder in einer Datei speichern möchten, wie im folgenden Beispiel umleiten:

```
$ sort /etc 2> /dev/null
```

Umleitung der Standardeingabe (Standard Input)

Diese Art der Umleitung dient dazu, einem Befehl Daten aus einer bestimmten Datei statt über die Tastatur zu übergeben. Das geschieht über den **Operator** `<`, wie im folgenden Beispiel zu sehen:

```
$ cat < text
Hello!
Hello to you too!
```

Die Umleitung der Standardeingabe wird normalerweise bei Befehlen verwendet, die keine Argumente akzeptieren. Der **Befehl** `tr` ist einer von diesen. Er dient dazu, **Dateiinhalte zu übersetzen**, indem man die Zeichen in einer Datei auf bestimmte Weise ändert, etwa durch das Löschen eines bestimmten Zeichens aus einer Datei. Das folgende Beispiel zeigt das Löschen des Zeichens `l`:

```
$ tr -d "l" < text
Heo!
Heo to you too!
```

Weitere Informationen finden Sie in der Man Page von `tr`.

Here Documents

Anders als bei den Ausgabeumleitungen verhält sich der **Operator** `<<`**.** Dieser Input Stream wird auch *here document* genannt. Ein *here document* repräsentiert einen Text- oder Code-Block, der an den Befehl oder das interaktive Programm übergeben werden kann. Verschiedene Skriptsprachen wie `bash`, `sh` und `csh` können **so Eingaben direkt von der Kommandozeile übernehmen, ohne Textdateien zu nutzen**.

Wie im folgenden Beispiel zu sehen, dient der Operator dazu, **Daten an den Befehl zu übergeben,** wobei das folgende Wort keinen Dateinamen bezeichnet. Das Wort wird vielmehr als Trennzeichen der Eingabe interpretiert und nicht als Inhalt berücksichtigt; daher wird es von `cat` nicht angezeigt:

```
$ cat << hello
> hey
> ola
> hello
```

```
hey
ola
```

Weitere Informationen finden Sie in der Man Page des Befehls `cat`.

Kombinationen

Die erste Kombination **leitet Standardausgabe und Standardfehlerausgabe** in eine **einzige Datei** um. Dazu nutzen wir die **Operatoren `&>` und `&>>`,** wobei `&` die **Kombination aus *Kanal 1* und *Kanal 2* repräsentiert**. Der erste Operator überschreibt den Inhalt einer vorhandenen Datei, der zweite fügt die neuen Informationen am Ende einer existierenden Datei hinzu. Beide Operatoren ermöglichen, wie in den vorangegangenen Abschnitten, die Erstellung der neuen Datei, wenn sie nicht bereits existiert:

```
$ find /usr admin &> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
$ find /etc/calendar &>> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
/etc/calendar
/etc/calendar/default
```

Schauen wir uns ein Beispiel mit dem Befehl `cut` an:

```
$ cut -f 3 -d "/" newfile
$ cat newfile

share
share
share
---------Omitted output----------
lib
games
find: `admin': No such file or directory
calendar
calendar
find: `admin': No such file or directory
```

Der Befehl `cut` schneidet mit der Option `-f` bestimmte Felder aus der Eingabedatei, in unserem Fall das dritte Feld. Damit der Befehl das Feld findet, ist zudem ein Trennzeichen mit der Option `-d` angegeben, in unserem Fall das Zeichen `/`.

Um mehr über den Befehl `cut` zu erfahren, konsultieren Sie die Man Page.

Pipes

Eine **Umleitung** wird meist verwendet, um **das Ergebnis eines Befehls** zu speichern und anschließend von einem **anderen Befehl verarbeiten zu lassen.** Diese Zwischenprozesse können sehr mühsam und kompliziert werden, wenn die Daten **mehrere Prozesse durchlaufen**. Um dies zu vermeiden, lassen sich Befehle direkt **über Pipes verknüpfen**. Die Ausgabe des ersten Befehls wird damit automatisch zur Eingabe des zweiten Befehls. Diese Verbindung wird über den **Operator** `|` (senkrechter Strich) hergestellt:

```
$ cat /etc/passwd | less
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
:
```

Im obigen Beispiel ändert der **Befehl** `less` nach dem Pipe-Operator die Art und Weise, wie die Datei angezeigt wird. Der Befehl `less` zeigt die Textdatei so an, dass der Benutzer eine **Zeile auf und ab scrollen kann**. `less` wird standardmäßig auch verwendet, um die Man Pages anzuzeigen, wie in den vorherigen Lektionen beschrieben.

Es ist möglich, **mehrere Pipes gleichzeitig** zu verwenden. Die **Zwischenbefehle**, die Eingaben empfangen, verarbeiten und dann eine Ausgabe erzeugen, **heißen *Filter***. Nehmen wir den Befehl `ls -l` und versuchen wir, die Anzahl der Wörter aus den ersten 10 Zeilen der Ausgabe zu zählen. Dazu nutzen wir den **Befehl** `head`**, der standardmäßig die ersten 10 Zeilen** einer Datei anzeigt, und dann den **Befehl** `wc`**, der die Wörter zählt:**

```
$ ls -l | head | wc -w
10
```

Wie bereits erwähnt, zeigt `head` standardmäßig nur die ersten 10 Zeilen der angegebenen Textdatei an. Mit Hilfe bestimmter Optionen ändert man dieses Verhalten. Weitere Informationen finden Sie in der Man Page des Befehls.

Es gibt daneben noch einen Befehl, der das Ende einer Datei anzeigt: `tail`. Standardmäßig wählt dieser Befehl **die letzten 10 Zeilen** aus und zeigt sie an. Aber wie bei `head` können Sie die Anzahl ändern (siehe die Man Page von `tail` für weitere Details).

> **Note** Die Option `-f` zeigt die letzten Zeilen einer Datei an, während sie aktualisiert wird. Diese Funktion ist sehr nützlich, wenn Sie eine Datei wie `syslog` auf laufende Aktivitäten überwachen.

Der **Befehl** `wc` **("word count") zählt standardmäßig die Zeilen**, Wörter und Bytes einer Datei. Wie in der Übung gezeigt, bewirkt die **Option** `-w`**, dass der Befehl nur die Wörter innerhalb der ausgewählten Zeilen zählt**. Die häufigsten Optionen, die Sie mit diesem Befehl verwenden, **sind** `-l` **(nur die Zeilen zählen) und** `-c` **(nur die Bytes zählen).** Weitere Optionen des Befehls sowie weitere Informationen über `wc` finden Sie in der Man Page.

Geführte Übungen

1. Listen Sie den Inhalt Ihres aktuellen Verzeichnisses einschließlich Eigentümer und Berechtigungen auf und leiten Sie die Ausgabe in eine Datei namens `contents.txt` in Ihrem Heimatverzeichnis um.
2. Sortieren Sie den Inhalt der Datei `contents.txt` aus Ihrem aktuellen Verzeichnis und fügen Sie ihn an das Ende einer neuen Datei namens `contents-sorted.txt` an.
3. Zeigen Sie die letzten 10 Zeilen der Datei `/etc/passwd` an und leiten Sie sie in eine neue Datei im Verzeichnis `Documents` Ihres Benutzers um.
4. Zählen Sie die Anzahl der Wörter in der Datei `contents.txt` und hängen Sie die Ausgabe an das Ende einer Datei `field2.txt` in Ihrem Heimatverzeichnis an. Sie müssen sowohl die Eingabe- als auch die Ausgabeumleitung verwenden.
5. Zeigen Sie die ersten 5 Zeilen der Datei `/etc/passwd` an und sortieren Sie die Ausgabe alphabetisch umgekehrt.
6. Zählen Sie mit der zuvor erstellten Datei `contents.txt` die Anzahl der Zeichen der letzten 9 Zeilen.
7. Zählen Sie die Anzahl der Dateien namens `test` im Verzeichnis `/usr/share` und dessen Unterverzeichnissen. Hinweis: Jede Zeilenausgabe des Befehls `find` steht für eine Datei.

Offene Übungen

1. Wählen Sie das zweite Feld der Datei `contents.txt` aus und leiten Sie die Standardausgabe und Fehlerausgabe in eine andere Datei namens `field1.txt` um.
2. Löschen Sie mithilfe des Eingabeumleitungsoperators und des Befehl `tr` die Bindestriche (`-`) aus der Datei `contents.txt`.
3. Worin besteht der größte Vorteil, nur Fehler in eine Datei umzuleiten?
4. Ersetzen Sie alle aufeinanderfolgenden Leerzeichen in der alphabetisch sortierten Datei `contents.txt` durch ein einziges Leerzeichen.
5. Eliminieren Sie in einer Befehlszeile die aufeinanderfolgenden Leerzeichen (wie in der vorangehenden Übung), wählen Sie das neunte Feld aus und sortieren Sie es umgekehrt alphabetisch und nicht case-sensitiv. Wie viele Pipes mussten Sie verwenden?

Zusammenfassung

In dieser Lektion haben Sie gelernt:

- Arten der Umleitung
- Wie Sie die Umleitungsoperatoren verwenden
- Wie Sie Pipes zum Filtern der Befehlsausgabe verwenden

Befehle, die in dieser Lektion verwendet wurden:

`cut:` Entfernt Abschnitte aus jeder Zeile einer Datei.

`cat:` Zeigt Dateien an oder verknüpft sie.

`find:` Sucht nach Dateien in einer Verzeichnishierarchie.

`less`: Zeigt eine Datei an, so dass der Benutzer zeilenweise scrollen kann.

`more:` Zeigt eine Datei seitenweise an.

`head:` Zeigt die ersten 10 Zeilen einer Datei an.

`tail:` Zeigt die letzten 10 Zeilen einer Datei an.

`sort:` Sortiert Dateien.

`wc:` Zählt standardmäßig die Zeilen, Wörter oder Bytes einer Datei.