Wiki / Umleitungen

Dieser Artikel wurde für die folgenden Ubuntu-Versionen getestet:

Dieser Artikel ist größtenteils für alle Ubuntu-Versionen gültig.

Inhaltsverzeichnis

Beim Arbeiten im Terminal bietet die **Bash** [http://wiki.ubuntuusers.de/Bash] verschiedene Möglichkeiten, die Ausgabe der einzelnen Befehle umzuleiten bzw. an andere Befehle weiterzuleiten. Ebenso besteht die Möglichkeit, dass Befehle, die standardmäßig von der Standardeingabe lesen, alternativ die Eingabe(n) aus einer Datei lesen.

Dieser Wiki-Beitrag bezieht sich primär auf die **Bash** [http://wiki.ubuntuusers.de/Bash]. Die Umleitungen > , < und | stehen in anderen POSIX-kompatiblen **Shells** [http://wiki.ubuntuusers.de/Shell] jedoch in gleicher Form zur Verfügung.

stdin, stdout, stderr - Kanäle der Bash

Vorab noch ein wenig Hintergrundwissen: Alle Befehle und Programme, welche in der Bash gestartet werden, erhalten drei Kanäle zugewiesen:

- Den Standardeingabekanal *stdin*, dieser hat die Nummer 0 (null). Normalerweise liest stdin Eingaben von der Tastatur, welche mit dem Terminal verbunden ist.
- Den Standardausgabekanal *stdout*, dieser hat die Nummer 1 (eins). Normalerweise schreibt stdout Ausgaben auf den Bildschirm, welcher mit dem Terminal verbunden ist.
- Den Standardfehlerkanal *stderr*, dieser hat die Nummer 2 (zwei). Normalerweise schreibt stderr Ausgaben auf den Bildschirm, welcher mit dem Terminal verbunden ist.

Umleiten der Ausgabe mit >

Mit Hilfe des "Größer als" Zeichens > lässt sich die Standardausgabe stdout umleiten. Eine typische (und sehr häufige) Anwendung ist das Umleiten in eine Datei. So schreibt zum Beispiel der Befehl

```
ls > verzeichnis.txt
```

den Inhalt des aktuellen Verzeichnisses in die Datei **verzeichnis.txt** anstatt in das Terminal.

Achtung!

Existiert die Datei **verzeichnis.txt** nicht, so wird diese angelegt. Existiert die Datei bereits, so wird diese ohne Rückfrage überschrieben!

Experten-Info:

Aus demselben Grund funktioniert beispielsweise auch sed s/muster/ersetzungstext/ < datei.txt > datei.txt nicht. Da die Öffnung des umgeleiteten Ausgabekanals - die damit einhergehende Löschung des Inhalts von **datei.txt** - der Öffnung des Eingabekanals vorausgeht, arbeitet sed mit einer leeren Datei.

Es ist aber auch möglich, die Ausgabe an existierende Dateien anzuhängen, in dem man statt ein **>** zwei **>>** verwendet. Der Befehl

```
ls ~/Desktop >> verzeichnis.txt
```

hängt den Inhalt von **~/Desktop** also an die Datei **verzeichnis.txt** an.

Neben stdout schreibt auch der Standardfehlerkanal stderr regelmäßig auf das Terminal. Möchte man die Ausgabe von stderr umleiten, muss man 2> statt > verwenden:

```
ls -la /home/user 2> fehler.txt
```

2> heißt soviel wie „leite den Kanal 2 (stderr) um“.

Hinweis:

Der Kanal 1 (stdout) muss bei der Umleitung nicht explizit benannt werden. ">" wird von der Bash automatisch mit "1>" gleichgesetzt.

Es lassen sich auch beide Kanäle gleichzeitig in zwei verschiedene Dateien umleiten:

```
ls -la > verzeichnis.txt 2> fehler.txt
```

So wird die Ausgabe von **ls** in die Datei **verzeichnis.txt** umgeleitet, Fehlermeldungen in die Datei **fehler.txt**. Weiterhin ist es auch möglich, beide Kanäle in eine Datei zu leiten:

```
ls -la > gemeinsam.txt 2>&1
```

2>&1 bedeutet also soviel wie „schreibe die Ausgabe von Kanal 2 (stderr) dorthin, wo die Ausgabe von Kanal 1 (stdout) geschrieben wird“.

Hinweis:

Die Reihenfolge ist wichtig: Es wird immer dorthin umgeleitet, wohin der angegebene Kanal gerade geleitet wird. Stellt man das 2>&1 also vor das > gemeinsam.txt, wird stderr auf die Standardausgabe geleitet, nur was ursprünglich auf die Standardausgabe geschrieben wurde, landet in der Datei.

Natürlich kann man auch bei Kanal 2 zwei größer-als-Zeichen verwenden. So lassen sich z.B. Fehlermeldungen während einer Datensicherung von Tar an eine Logdatei anhängen:

```
echo "Sicherung" >> backup.log
tar -cf sicherung.tgz "/home" 2>> backup.log
```

Die Verwendung von &> leitet sowohl stdout als auch stderr in die angegebene Datei um. Das ist hilfreich u.a. in Bash-Skripten, in denen lediglich der Rückgabewert von Befehlen interessiert, nicht aber der Ausgabetext. Dieses Beispiel beendet ein Skript, wenn irgendeine Datei im **Homeverzeichnis** [http://wiki.ubuntuusers.de/Homeverzeichnis] geöffnet ist und unterdrückt dabei jegliche Ausgaben:

```
1 if lsof ~ &>/dev/null; then
2  exit 1;
3 fi
```

Die Umleitung &>> funktioniert hingegen *nicht*. Anstelle dessen muss man stdout an eine Datei anhängen und stderr wie oben beschrieben an stdout binden. Dieses Beispiel führt ein Kommando aus und hängt sowohl stdout als auch stderr an ein logfile:

```
./my_command.sh >> results.log 2>&1
```

Umleitung der Eingabe mit <

Mit Hilfe des „Kleiner als“-Zeichens < lässt sich die Standardeingabe (stdin) umleiten. Beispiel:

```
tr -d '0-9' < datei.txt
```

Dieser Aufruf zeigt den Inhalt aus **datei.txt** ohne Ziffern an.

Der Pipe-Operator |

Der Pipe-Operator (Pipe = Kurzform für Pipeline) leitet die Ausgabe eines Befehls direkt an einen anderen Befehl weiter (anstatt ins Terminal). Damit kann der zweite Befehl das Ergebnis bzw. die Ausgabe des ersten Befehls weiterverarbeiten. Die allgemeine Syntax lautet (man kann natürlich auch mehr als zwei Befehle miteinander verbinden):

```
Befehl1 | Befehl2
```

Hinweis:

Das Zeichen für den Pipe-Operator | erhält man auf einer deutschen Tastatur durch Drücken von Alt Gr + < .

Eine typische Anwendung für den Pipe-Operator ist z.B. das Ausrufen eines Befehls, der eine größere Menge an Daten auf stdout schreibt (z.B. Ausgaben von Systemmeldungen wie dmesg oder Ausgaben von Prozessinformationen wie **ps** [http://wiki.ubuntuusers.de/Shell/ps] und **pstree** [http://wiki.ubuntuusers.de/Shell/pstree]) in Kombination mit Datensortierung (z.B. sort) oder Durchsuchen der Ausgabe nach bestimmten Ausdrücken (z.B. **grep** [http://wiki.ubuntuusers.de/Shell/grep]). Einige Beispiele:

- Im ersten Beispiel werden alle laufenden Prozesse durch ps ausgegeben, der Pipe-Operator leitet die Ausgabe an sort weiter. sort sortiert die Daten um (absteigende numerische Sortierung, Option -nr) und gibt die umsortierten Daten dann an stdout weiter.

```
ps ax | sort -nr
```

- Im zweiten Beispiel gibt dmesg alle Log-Meldungen des Kernels aus, der Pipe-Operator leitet diese an grep weiter. grep sucht nur nach Zeilen, in denen der Ausdruck usb vorkommt. Die Ausgabe von grep erfolgt dann (standardmäßig) auf stdout (der Parameter -n sorgt dafür, dass grep die Zeilen nummeriert).

```
dmesg | grep -n USB
```

- Im dritten Beispiel kommen mehrere Pipelines zum Einsatz. Das zweite Beispiel wird durch einen weiteren Pipe-Operator und den Befehl tail ergänzt. Somit werden nur noch die letzten zehn Zeilen angezeigt.

```
dmesg | grep -n USB | tail
```

- Natürlich lässt sich der Pipe-Operator auch mit der Umleitung von stdin und stdout kombinieren, zum Beispiel:

```
grep usb < alle-meldungen.log | tail > usb-meldungen.log
```

Hinweis:

Die Umleitungen müssen übrigens nicht hinter dem Befehl stehen, sie können auch davor oder mittendrin geschrieben werden:

```
grep < alle-meldungen.log usb | >usb-meldungen.log tail
```

tee - Ausgabe verdoppeln

Das Programm **tee** liest von der Standardeingabe stdin und verdoppelt die eingelesenen Daten. Die Daten werden dann je einmal an eine Datei und an die Standardausgabe stdout weitergeleitet. tee ist im essentiellen Paket

- **coreutils**

von Ubuntu enthalten und ist deshalb auf jedem System installiert.

Die allgemeine Syntax lautet:

```
tee <Optionen> Ausgabedatei
```

tee kennt die folgenden Optionen:

tee - Optionen	
Option	**Beschreibung**
-a oder --append	Die Daten aus der Standardeingabe stdin werden an die Ausgabedatei angehängt, anstatt diese zu überschreiben.
-i oder --ignore-interrupts	Interrupt-Signale werden ignoriert.
--help	Zeigt Hilfsinformationen zu tee an.
--version	Zeigt die Versionsnummer von tee an.

Üblicherweise wird tee in einer Pipe verwendet, wie z.B.:

```
ls -la | tee alle_dateien.txt | grep '\.png$'
```

Es wird das ausführliche Inhaltsverzeichnis angezeigt, einmal (vollständig) in die Datei **alle_dateien.txt** geschrieben und einmal an grep weitergeleitet, welches nur alle **png**-Dateien auf stdout ausgibt.

Links

- **script** [http://wiki.ubuntuusers.de/script] - Terminalsitzung mitschneiden
- **Shell/Befehlsübersicht** [http://wiki.ubuntuusers.de/Shell/Befehls%C3%BCbersicht] Übersicht über verschiedene Shell-Befehle

---

Diese Revision [http://wiki.ubuntuusers.de/Shell/Umleitungen?rev=732957] wurde am 18. Juni 2014 11:34 von **WinXP to Edgy** erstellt.
Die folgenden Schlagworte wurden dem Artikel zugewiesen: **Shell** [http://wiki.ubuntuusers.de/Wiki/Tags?tag=Shell]