Pipes, Umleitungen und Text-Befehle

ITS-Net-Lin

Sebastian Meisel

13. Dezember 2024

1 Pipes und Text-Befehle

In Linux und anderen Unix-ähnlichen Betriebssystemen können Pipes verwendet werden, um die Ausgabe eines Befehls direkt als Eingabe für einen anderen Befehl zu nutzen. Pipes ermöglichen es, mehrere Befehle miteinander zu verketten und so komplexe Aufgaben effizient und flexibel zu lösen.

1.1 Pipes in der Praxis

Eine Pipe wird durch das Symbol | dargestellt und verbindet die Ausgabe eines Befehls mit der Eingabe eines anderen. Zum Beispiel:

```
ls -l | grep "txt"
```

In diesem Beispiel listet der Befehl `ls -l` die Dateien im aktuellen Verzeichnis auf, und die Ausgabe wird an `grep` weitergeleitet, das nur die Zeilen mit dem Text `txt"` anzeigt.

2 Umleitungen (Redirections)

2.1 |&

Diese Umleitung kombiniert Standardausgabe (stdout) und Standardfehlerausgabe (stderr) und leitet beide in denselben Zielstrom um. Beispiel:

```
find / -name test |& less
```

2.2 <

Mit dieser Umleitung wird eine Datei oder ein Datenstrom als Eingabe für einen Befehl verwendet. Beispiel:

```
cat < input.txt
```

Das hat denselben Effekt wie `cat input.txt` (ohne Umleitung), funktioniert aber auch dann, wenn der Befehl keine Dateien lesen kann.

2.3 <<<

Hierbei handelt es sich um eine sogenannte „Here-String"-Umleitung, bei der ein String direkt als Eingabe an einen Befehl übergeben wird. Beispiel:

```
grep 'pattern' <<< "This␣is␣a␣test␣string"
```

Das ist im Prinzip das Gegenteil von <: Ich kann damit einen String an einen Befehl übergeben, der eine Datei als Eingabe erwartet.

2.4 >

Diese Umleitung schreibt die Standardausgabe (stdout) in eine Datei oder einen anderen Zielstrom (meist in eine Datei). Beispiel:

```
ls > output.txt
```

2.5 >>

Dies hängt die Ausgabe eines Befehls an das Ende einer Datei an, anstatt sie zu überschreiben. Beispiel:

```
ls >> output.log
```

2.6 2>

Diese Umleitung schreibt ausschließlich die Standardfehlerausgabe (stderr) in eine Datei oder einen Zielstrom. Beispiel:

```
find / -name test 2> error.log
```

2.7 2>&1

Mit dieser Umleitung wird stderr auf stdout umgeleitet, sodass beide Ausgaben kombiniert werden können. Beispiel:

```
find / -name test 2>&1 | tee combined.log
```

2.8 &>

Diese Umleitung macht genau dasselbe wie `2>&1`, nur in einer kürzeren Form - funktioniert aber nur in Bash und Zsh, aber in vielen anderen Shells nicht.

```
command >& output.log
```

3 Wichtige Befehle

`cat` Der Befehl `cat` wird häufig verwendet, um den Inhalt von Dateien anzuzeigen oder mehrere Dateien zusammenzuführen. Ein einfaches Beispiel ist:

```
cat datei.txt
```

`grep` `grep` wird verwendet, um nach einem bestimmten Muster in einer Datei oder der Ausgabe eines Befehls zu suchen. Beispiel:

```
cat datei.txt | grep "Suchbegriff"
```

`sort` Mit `sort` können die Zeilen einer Datei oder Eingabe alphabetisch oder numerisch sortiert werden. Beispiel:

```
cat datei.txt | sort
```

3.1 Verwendung von `more` und `less`

Die Befehle `more` und `less` ermöglichen das schrittweise Anzeigen von Inhalten, insbesondere bei langen Texten, die nicht vollständig in das Terminal passen.

- **`more`:** Der Befehl `more` zeigt den Inhalt einer Datei seitenweise an. Mit der Leertaste kann man zur nächsten Seite wechseln, und mit der Taste q beendet man die Anzeige. Beispiel:

```
more datei.txt
```

- **`less`:** `less` ist eine erweiterte Version von `more` und bietet zusätzliche Navigationsmöglichkeiten. Mit den Pfeiltasten oder der Bildlauf-Funktion kann man vor- und zurückscrollen. Mit q beendet man auch hier die Ansicht. Beispiel:

```
less datei.txt
```

`less` ist besonders nützlich, da es den gesamten Inhalt vorab lädt, was die Navigation in sehr großen Dateien erleichtert.

3.2 Ersetzen von Text mit `sed`

Der Befehl `sed` ist ein Stream-Editor, der es ermöglicht, Text in der Eingabe zu bearbeiten. Ein häufiger Anwendungsfall ist die Ersetzung von Mustern.

- Einmalige Ersetzung: : Um ein Muster einmal zu ersetzen, verwendet man:

```
    sed 's/pattern/replace/' datei.txt
```

  Dieser Befehl ersetzt das erste Vorkommen von ‚pattern' mit ‚replace' in jeder Zeile.

- Globale Ersetzung: : Mit der Option g kann man alle Vorkommen in einer Zeile ersetzen:

```
    sed 's/pattern/replace/g' datei.txt
```

  Dieser Befehl ersetzt jedes Vorkommen von ‚pattern' mit ‚replace' in jeder Zeile.

ilen adressieren Man kann mit `sed` auch bestimmte Zeilen ansprechen, um dort Text zu ersetzen:

1,4 Ersetzt in den Zeilen 1 bis 4:

```
      sed '1,4s/pattern/replace/g' datei.txt
```

3,$ Ersetzt von Zeile 3 bis zum Ende der Datei:

```
      sed '3,$s/pattern/replace/g' datei.txt
```

5,+4 Ersetzt in Zeile 5 und den nächsten 4 Zeilen:

```
      sed '5,+4s/pattern/replace/g' datei.txt
```

***pattern*,/pattern/** Ersetzt zwischen den Zeilen, die mit dem ersten und dem zweiten ‚pattern' übereinstimmen:

```
      sed '/pattern1/,/pattern2/s/pattern/replace/g' datei.txt
```

Eine weitergehende Einführung bietet mein Tool sedtutor. Dieses können Sie unter Windows herunterladen und dann wie folgt auf die Linux-VM transportieren:

```
    cd ~\Downloads
    scp sedtutor.txt debian:~/sedtutor
```

In der Linux-VM müssen Sie nun im Terminal die Ausführungsrechte für Ihren Nutzer setzen:

```
chmod u+x sedtutor
```

Nun können Sie das Skript wie folgt ausführen:

```
./sedtutor
```

Folgen Sie den Anweisungen. Mit `[Q]` können Sie das Skript jederzeit beenden um später fortzusetzen.

3.3 Beispiel für die Verwendung von Pipes mit mehreren Befehlen

Angenommen, wir möchten eine Datei nach einem bestimmten Muster durchsuchen, das Ergebnis sortieren und dann die Duplikate entfernen:

```
cat datei.txt | grep 'Muster' | sort | uniq
```

Hierbei wird der Inhalt von `datei.txt` nach dem Muster ‚Muster' durchsucht, dann nach alphabetischer Reihenfolge sortiert und schließlich werden doppelte Zeilen entfernt.

3.4 Fazit

Pipes und die Kombination von Befehlen wie `cat`, `grep`, `sort` und `sed` bieten leistungsstarke Möglichkeiten zur Verarbeitung von Textdaten in Linux. Durch das Erlernen und Anwenden dieser Werkzeuge kann man Daten effizient durchsuchen, filtern und bearbeiten.