# 12. Pipes und andere Besonderheiten

## Ein− und Ausgabe−Umlenkung

**UNIX: Small is beautiful.**

**Es existieren viele kleine Programme, die jeweils eine kleine Aufgabe sehr gut machen. Eine große Aufgabe wird durch die Nacheinanderausführung von verschieden kleinen Programmen gelöst.**

**Problem: Wie erfolgt der Datentransport von einem Programm zum anderen?**

**Jedes Programm hat bei seinem Start drei E/A−Kanäle geöffnet, die es sofort benutzen kann.**

| | | |
|---|---|---|
| **Standardeingabe** | **−** | **Eingabekanal für Daten** |
| **Standardausgabe** | **−** | **Ausgabekanal für Ergebnisse** |
| **Standarderror** | **−** | **Ausgabekanal für Fehlertexte** |

**Die E/A−Kanäle werden von den Shells verwaltet (wenn die Kommandos von der Shell aus gestartet werden).**
**Standardmäßig legen die Shells die Standardeingabe auf die Tastatur.**
**Standardausgabe und Standarderror werden von den Shells auf den Bildschirm gelegt.**

## Umlenkung von Standardausgabe und Standardeingabe

Standardeingabe und Standardausgabe können von den Shells auf Files umgelenkt werden, so daß ein Programm dann die Eingabedaten aus einem File bzw. die Ausgabedaten in ein File schreibt.

`>` − Umlenkung der Ausgabedaten

```
> <Filename des Ausgabefiles>
```

Achtung!!! Das Ausgabefile wird ohne Warnung überschrieben, wenn es bereits existiert.

`<` − Umlenkung der Eingabedaten

```
< <Filename des Eingabefiles>
```

Das Eingabefile muß existieren.

Beispiel:

```
ls −lisa > xxx
more < xxx
```

Die Programm more, less, tail, od lesen ihre Eingabe von Standardeingabe, wenn kein Filename angegeben wurde.

## Umlenkung von Standardfehlerausgabe

Umlenkung der Standardfehlerausgabe kann nur bei Bourne Shell, Bash und Kornshell erfolgen.

```
2> <Ausgabefilename>
```

Soll bei einer Ausgabeumlenkung das Ausgabefile nicht überschrieben werden, ist folgendes zu notieren:

```
>> <Ausgabefilename>
```

bzw.

```
2>> <Ausgabefilename>
```

## Die Pipe im Unix

Die Pipe dient der Verkettung von Programmen. Dabei wird die Standard−ausgabe des ersten Programms mit der Standardeingabe des zweiten Programms verbunden. Beide Programme werden parallel gestartet und laufen dann anschließend parallel ab, so daß die Ausgabedaten des ersten Programms direkt als Eingabedaten vom zweiten Programm genutzt werden können.

Bei der Benutzung der Shells wird eine Pipe mittels

"|"

kodiert.

```
<Kommando 1> | <Kommando 2>
```

Natürlich kann man beliebig viele Kommandos über mehrere Pipes verbinden.

```
<Kommando 1> | <Kommando 2> | <Kommando 3> | <Kommando 4> | ....
```

Beispiel:

```
ls −lisa
ls −lisa | more
ls −lisa | grep 2003−10 | more     # Linux
ls −lisa | grep Oct | more         # Solaris
```