Linux von Anfang an, zweiter Teil

Lian Herzberg

05.12.2025

3 Schnittstellen auf der Kommandozeile

3.1 Tastenkürzel auf der Kommandozeile

Mit Tastenkürzeln (Shortcuts) können wir auf der Kommandozeile schneller arbeiten. Einige Funktionen sind nur über diese Tastenkombinationen zugänglich.

Strg-C bricht einen laufenden Befehl ab (C wie cancel). Das funktioniert bei allen Kommandos, die nicht den gesamten Bildschirm vereinnahmen.

Strg-L bewirkt dasselbe wie das Kommando `clear`. Der Bildschirm wird aufgeräumt, vorherige Inhalte sind durch Scrollen noch erreichbar. Das Kommando `reset` bewirkt ein vollständiges Zurücksetzen des Terminals auf seinen Ausgangszustand. Das kann uns auf der Konsole außerhalb der grafischen Oberfläche helfen, wenn durch bestimmte Ausgaben das Terminal eine fehlerhafte Anzeige aufweist.

Strg-D oder der Befehl `exit` bewirken einen Logout. Manche Programme werden mit dieser Tastenkombination verlassen.

Pfeil hoch und Pfeil runter können benutzt werden, um vorangegangene Kommandos durchzuschalten. Der Verlauf der eingegebenen Befehle wird je nach Shell unterschiedlich gespeichert. Die Bash legt die Datei `.bash_history` im Home-Verzeichnis des aktuellen Users an.

Tab wird zum Vervollständigen von Befehlen und Pfaden benutzt (die sog. Tab-Completion). Wenn eine angefangene Eingabe eindeutig ist, vervollständigt Tab den eingegebenen Befehl oder Pfad. Wenn die Eingabe nicht eindeutig ist, unterscheidet sich das Verhalten zwischen den verschiedenen Shells. In der Bash kann erneut Tab gedrückt werden, um sich eine Liste der Möglichkeiten anzeigen zu lassen.

Shift-Bild Hoch und Shift-Bild Runter helfen dabei, im Terminal nach oben und unten zu scrollen, ohne die Maus benutzen zu müssen. Manchmal funktionieren diese Tastenkombinationen auch auf der Konsole (außerhalb der grafischen Oberfläche).

3.2 Standard-Datenströme

Die Standard-Ausgabe (stdout) ist der Name für die Textausgabe, die von einem Befehl ausgeht. Wir lernen ein neues Kommando kennen:

```
> echo hallo
```

Das Kommando `echo` gibt eine übergebene Zeichenkette (String) auf die Standard-Ausgabe aus. Im Normalfall bedeutet das: direkt auf den Bildschirm.

Die Standardausgabe kann in eine Datei umgeleitet werden.

```
> echo hallo > hallo.txt
```

Die schließende spitze Klammer schreibt den Inhalt der Standardausgabe des Befehls in eine Datei. Wir können beobachten, dass dieser Befehl uns keine Ausgabe anzeigt, obwohl `echo` uns die Zeichenkette normalerweise direkt ausgeben würde.

```
> cat hallo.txt
```

Mit dem Kommando `cat` wird der Inhalt von Dateien auf die Standardausgabe gegeben. Die Man-Page von `cat` beschreibt die Funktionalität im Detail. Das funktioniert mit allen Dateien, nicht nur mit Text-Dateien, allerdings sehen Binärdateien unter Umständen sehr eigenartig aus, wenn man sie als Text ausgeben lässt.

Wir sehen, dass in der Datei die Zeichenkette steht, die der `echo`-Befehl ausgegeben hat. Die Standard-Ausgabe des Befehls wurde in die Datei umgeleitet.

```
> echo moin > hallo.txt
> cat hallo.txt
```

Nach dem Eingeben dieser beiden Befehle sehen wir, dass der Inhalt der Text-Datei überschrieben wurde. Die einfache schließende spitze Klammer überschreibt immer die komplette Datei, egal, wie lang sie vorher war, mit der neuen Ausgabe.

```
> echo salve >> hallo.txt
> cat hallo.txt
```

Wenn wir statt einer zwei schließende spitze Klammern verwenden, wird die Standardausgabe des vorherigen Befehls an die angegebene Datei angehängt. Mit Pfeil hoch können wir den `echo`-Befehl mehrmals wiederholen und erneut ausführen, um zu demonstrieren, dass die Datei tatsächlich länger wird.

Die Standardeingabe (stdin) erwartet üblicherweise eine Tastatureingabe. Wer die Kommandos `sudo` oder `su` schon einmal benutzt hat, wurde dabei sicherlich nach einem Passwort gefragt und hat dieses über die Standardeingabe eingegeben. Auch wenn beim Installieren von Paketen über die Kommandozeile nach einer Bestätigung gefragt wird und ein Y oder N eingegeben werden soll, erfolgt die Verarbeitung dieser Eingabe über die Standardeingabe.

Die Pipe dient dazu, die Standardausgabe von einem Befehl als Standareingabe des nächsten Befehls zu verwenden. Die Funktionsweise lässt sich mit unserer längeren Textdatei und einem neuen Kommando lernen:

```
> cat hallo.txt | less
```

Die Ausgabe des `cat`-Befehls wird dem `less`-Befehl als Eingabe übergeben. `less` ist ein Programm, das dem Anzeigen von Textdateien dient. Standardmäßig werden Man-Pages in `less` angezeigt, sodass uns die Funktionsweise des Programms bereits bekannt ist. Mit der Taste `Q` kann `less` verlassen werden.

An dieser Stelle die Pipe zu verwenden, ist zugegebenermaßen nicht besonders sinnvoll, da mit `less hallo.txt` direkt die Textdatei angezeigt werden kann. Allerdings befindet sich nicht jeder lange Text-Output, den wir vielleicht bequem anschauen und durchsuchen möchten, in einer lesbaren Textdatei.

Die Standardfehlerausgabe (stderr) unterscheidet sich nur im Detail von der Standardausgabe. Wir lernen dies, indem wir absichtlich eine Fehlermeldung herbeiführen, etwa indem wir mit `cd` in ein Verzeichnis zu wechseln versuchen, das nicht existiert:

```
> cd gibtsnicht
```

Um zu demonstrieren, dass es sich bei diesem Output nicht um eine Standardausgabe handelt, leiten wir die Standardausgabe in eine Datei um:

```
> cd gibtsnicht > output.txt
```

Wir sehen, dass die Fehlermeldung weiterhin angezeigt wird und nicht in der Datei landet. Wenn wir uns den Inhalt der Datei ausgeben lassen, stellen wir fest, dass diese leer ist. Die Fehlermeldung wurde nämlich nicht als Standardausgabe übergeben, sondern als Standardfehlerausgabe.

3.3 Die Shell

Eine Shell ist ein Programm mit dem Zweck, Ein- und Ausgabe zu ermöglichen. Sie ist eine Bedienschnittstelle (User Interface, UI) oder genauer gesagt: eine Kommandozeilenschnittstelle (Command Line Interface, CLI).

Auf der grafischen Oberfläche starten wir ein sogenanntes Terminal, um auf die Kommandozeile zuzugreifen. Dieses Terminal stellt ein Fenster zur Verfügung, innerhalb dessen die Shell ausgeführt wird.

Das Programm, das unsere eingegebenen Befehle verarbeitet, uns die Standardausgaben von Befehlen anzeigt und unsere Tastatureingaben an die ausgeführten Programme schickt, ist die Shell.

Es gibt viele verschiedene Shells, die sich in ihrer Bedienung im Detail unterscheiden. In jeder Shell werden die gleichen Befehle verwendet. Die Befehle sind eigenständige Programme und existieren unabhängig von der verwendeten Shell. Nur einige wenige Funktionen (z.B. das Kommando `cd`) sind Teil der Shell selbst.

Unterschiedlich zwischen verschiedenen Shells sind etwa die Darstellung von Textinhalten, das Verhalten der Tab-Completion, der Umgang mit dem Befehlsverlauf oder die Konfigurationsmöglichkeiten.

Standardmäßig unter Linux wird die Bash verwendet. Andere Shells können je nach eigenen Nutzungsvorlieben die Arbeit auf der Kommandozeile stark erleichtern.

3.4 Variablen in der Shell

Eine Shell kann auch Variablen speichern. Wie in der Mathematik sind Variablen in Programmiersprachen ein Behälter für einen Wert. Der Name einer Variablen bleibt immer gleich, während ihr Inhalt sich ändern kann.

Einige Variablen sind in der Shell standardmäßig gesetzt. Wir können uns den Inhalt einer solchen Standard-Variable anzeigen lassen:

```
> echo $SHELL
```

Mit diesem Befehl wird der Inhalt der `SHELL`-Variable ausgegeben. Variablen beginnen mit dem `$`-Zeichen.

Umgebungsvariablen sind Variablen, die das Verhalten von Programmen beeinflussen, Informationen über die Umgebung enthalten oder auch Standardprogramme definieren. Die `SHELL`-Variable ist eine solche Umgebungsvariable. Sie kann von Programmen abgerufen werden, wenn sie z.B. ihr Verhalten an die verwendete Shell anpassen möchten.

Sämtliche gesetzten Umgebungsvariablen können mit dem Befehl `env` angezeigt werden, und hier lässt sich auch gleich demonstrieren, wie praktisch die Pipe ist, denn mit ihrer Hilfe können wir uns die Ausgabe in `less` ansehen, statt im Terminal scrollen zu müssen:

```
> env | less
```

Eine weitere Umgebungsvariable ist `EDITOR`, womit der Standard-Texteditor für die Kommandozeile definiert wird. (Wenn diese Variable nicht gesetzt wird, landet man häufig in `vi`, der als Texteditor alles andere als anfängerfreundlich ist!)

```
> echo $EDITOR
```

Bei beiden Variablen fällt uns auf, dass sie einen Pfad zu einer Datei enthalten. Mit dem bekannten Befehl `ls -l /pfad/zur/datei` können wir uns Details zu diesen Dateien anzeigen lassen.

In den meisten Fällen werden sich die beiden Dateien in `/usr/bin/` befinden. Wir können uns anschauen, was in diesem Verzeichnis sonst noch liegt:

```
> ls -l /usr/bin
```

Wenn wir ein wenig scrollen, werden wir dort einige Anwendungen oder Kommandos wiederfinden, die uns bekannt vorkommen, zum Beispiel eine Datei namens `firefox` oder eine namens `echo`.

Die Pfad-Variable ist eine Umgebungsvariable, in der sämtliche Verzeichnisse aufgelistet werden, in denen nach Befehlen gesucht werden soll.

```
> echo $PATH
```

Ein Befehl auf der Kommandozeile ist nichts anderes als eine ausführbare Datei, die in einem in `$PATH` aufgelisteten Verzeichnis aufzufinden ist. Auch jedes installierte Programm ist nichts anderes als eine ausführbare Datei, das in einem solchen Verzeichnis liegt.

Unter Linux wird hierzu heutzutage meist nur noch `/usr/bin` verwendet. Dort liegen alle eure installierten Programme wie Texteditor, die Shell selbst, Webbrowser, Bildbearbeitungsprogramme u.v.m.

Die einzige Ausnahme sind die bereits schon erwähnten Befehle, die Teil der Shell sind und deren Grundfunktionalität herstellen. (Dazu gehören neben `cd` auch noch Steuerungskommandos, zum Beispiel die Schleifenbefehle `while` und `for` oder der Bedingungsbefehl `if`.)

Verallgemeinert lässt sich sagen: Installierte Programme sind ausführbare Dateien, die an bestimmten Orten im Dateisystem liegen oder anderweitig im Betriebssystem registriert sind.