Erste Schritte mit der Kommandozeile

Die Kommandozeile (auch Konsole oder Terminal) ist ein textbasierter Eingabebereich (Interface) zum Ausführen von Befehlen auf einem Computer. Der Kommandozeileninterpreter zeigt dir eine Eingabeaufforderung (auch Prompt) an, hinter die du einen Befehl schreiben und mit „ENTER“ ausführen kannst.

```
/Dokumente$   touch Neu.txt
(=Prompt)     (=Befehl)
```

Los geht’s

Nach dem Öffnen der Kommandozeile zeigt dir der Prompt außerdem an, in welchem Verzeichnis du dich aktuell befindest. Das ist wichtig, da die eingegebenen Befehle immer von einem bestimmten Verzeichnis aus ausgeführt werden. Wenn ein Befehl einen Dateiname oder ein Verzeichnis enthält kannst du dieses auf zwei verschiedenen Arten spezifizieren.

Relativer Pfad

Ein **relativer Pfad** gibt an, wo ein Verzeichnis liegt relativ zu dem Verzeichnis, in dem du dich befindest. Bist du zum Beispiel im Unterordner „Videos“ im Verzeichnis „Dokumente“, dann zeigt der Prompt dies an:

```
/Dokumente/Videos$
```

Wenn du den Befehl `touch Neu.txt` ausführst wird im Unterordner „Videos“ eine neue Textdatei mit dem Name „Neu.txt“ angelegt.

Absoluter Pfad

Ein **absoluter Pfad** gibt an, wo sich ein Verzeichnis absolut betrachtet auf dem Computer befindet. Befindet du sich zum Beispiel wieder im Unterordner „Videos“ im Verzeichnis „Dokumente“ mit dem Prompt:

```
/Dokumente/Videos$
```

und führst den Befehl `touch /Dokumente/Musik/Neu.txt` aus, dann wird im Unterordner „Musik“ im Verzeichnis „Dokumente“ eine neue Textdatei mit dem Namen „Neu.txt“ angelegt.

Befehle mit absoluten Pfaden sind also unabhängig vom Verzeichnis, in dem du dich befindest und liefern immer das gleiche Ergebnis. Im Gegensatz dazu sind Befehle mit relativen Pfaden abhängig vom aktuellen Verzeichnis und liefern je nach Prompt unterschiedliche Ergebnisse.

Diese zwei Befehle liefern zum Beispiel das gleiche Ergebnis, wenn sie in /Dokumente/Videos ausgeführt werden:

```
/Dokumente/Videos$   rm Video.mp4
(Lö scht„ Video.mp4  aus dem aktuellen Verzeichnis)

/Dokumente/Videos$   rm /Dokumente/Videos/Video.mp4
(Lö scht „ Video.mp4   aus  dem  Verzeichnis  /Dokumente/Videos,  welches  zufä llig  das
aktuelle Verzeichnis ist)
```

Aus einem anderen Verzeichnis liefern sie jedoch unterschiedliche Ergebnisse:

```
/Dokumente/Musik$    rm Video.mp4
(Versucht „Video.mp4“ aus dem aktuellen Verzeichnis also dem Unterordner „Musik“ lö schen,
da dies das aktuelle Verzeichnis ist)

/Dokumente/Musik$    rm /Dokumente/Videos/Video.mp4
(Lö scht „ Video.mp4   aus  dem  Verzeichnis  /Dokumente/Videos,  auch  wenn  es  nicht  das
aktuelle Verzeichnis ist)
```

Merke: Wenn du den ganzen Pfad angeben willst und das aktuelle Verzeichnis ignorieren willst, beginne den Pfad mit einem Slash (/). Wenn du von deinem aktuellen Verzeichnis aus weiter navigieren willst, setze keinen Slash an den Anfang des Pfades.
Den absoluten Pfade deines aktuellen Verzeichnisses erhältst du mit dem Befehl `pwd`.

Muster

Oft kannst du statt einem konkreten Datei- oder Verzeichnisname auch ein **Muster** angeben, das auf mehrere Dateien passt. Es gibt viele Möglichkeiten solche Muster anzugeben, wobei die einfachste der Asterisk (*) ist, welcher für alles beliebige steht.

Löscht alle Dateien im aktuellen Verzeichnis:

```
/Dokumente$   rm *
```

Löschte alle Dateien, die auf „.txt“ enden:

```
/Dokumente$   rm *.txt
```

Löscht alle Dateien, die mit „Neu“ anfangen:

```
/Dokumente$   rm Neu*
```

Navigation

Die zwei wichtigsten Befehle zum navigieren im Verzeichnis sind `cd` und `ls`.

Mit dem Befehl `cd` kannst du das aktuelle Verzeichnis (**c**urrent **d**irectory) wechseln. Du kannst einen absoluten oder einen relativen Pfad angeben.

Wechselt zu /Dokumente /Videos:

```
/Dokumente$   cd Videos
/Dokumente/Videos$
```

Wechselt zu /Downloads und dann in den Unterordner /Arbeit

```
/Dokumente$   cd /Downloads
/Downloads$   cd Arbeit
/Downloads /Arbeit$
```

Du kannst auch mehrere Ebenen auf einmal springen

```
/Dokumente$   cd Videos /Lustig
/Dokumente / Videos /Lustig$
```

Mit cd .. gehst du eine Ebene nach oben:

```
/Dokumente/Videos /Lustig$  cd ..
/Dokumente /Videos$
```

Mit dem Befehl `ls` kannst du dir anzeigen lassen, welche Dateien und Ordner sich im aktuellen Verzeichnis befinden.

```
/Dokumente /Videos$  ls
Lustig        Video1.mp4          Video2.mp4
```

Mit der Taste „Tab“ kannst du, einen Datei- oder Verzeichnisname vervollständigen lassen. Tippst du zum Beispiel:

```
/Dokumente / Videos$ cd Lu
```

und dann „Tab“, wird die Zeile ergänzt zu:

```
/Dokumente /Videos$ cd Lustig
```

Wenn die Eingabe nicht eindeutig ist, kann der Name nicht ergänzt werden.
Beispiel:

```
/Dokumente / Videos$ cd Vi
```

und dann „Tab“, wird die Zeile ergänzt zu:

```
/Dokumente /Videos$ cd Video
```

Drückst du nun erneut „Tab“, so passiert nichts, da es zwei verschiedenen Möglichkeiten gibt, den Name zu vervollständigen (Video1.mp4 oder Video2.mp4).

Ausgabe

Die folgenden Befehle liefern jeweils Text zurück. Befehl `head [Datei]` liefert zum Beispiel die ersten 10 Zeilen der angegebenen Dateien. Tippst du den Befehl ein und drückst „Enter“, dann werden die Zeilen in der Konsole angezeigt.

```
/Dokumente$   head Wunderlinge.txt
Dr. Henry Pym
Felicia Hardy
Natascha Romanoff
James Buchanan Barnes
Steven Rogers
Matthew Michael Murdock
Ben Grimm
Wade Winston Wilson
Jonathan Storm
Susan Storm
/Dokumente$
```

Wenn du dir den Inhalt einer Datei nur anschauen möchtest, ist dies sehr praktisch. Du kannst aber auch die Ausgabe in einer Datei speichern oder an einen anderen Befehl weitergeben.

Ausgabe in einer Datei speichern

```
/Dokumente$   head Wunderlinge.txt > Erste10Wunderlinge.txt
```

Wenn „Erste10Wunderlinge.txt“ noch nicht existiert, wird die Datei erzeugt, ansonsten überschrieben.

```
/Dokumente$   head Wunderlinge.txt >> MehrWunderlinge.txt
```

Fügt die Ausgabe am Ende an die Datei „MehrWunderlinge.txt“ an.

Ausgabe an einen anderen Befehl übergeben

Du kannst mithilfe des Pipe Symbols (|) eine Ausgabe an einen andern Befehl weitergeben. So kannst du zum Beispiel mit `grep` (mehr dazu später) in den ersten 10 Zeilen von „Wunderlinge.txt“ nach Steven suchen:

```
/Dokumente$ head Wunderlinge.txt | grep „Steven“
```

Dieser Befehl liefert das gleiche Ergebnis wie:

```
/Dokumente$ head Wunderlinge.txt > Erste10Wunderlinge.txt
/Dokumente$ grep „Steve“ Erste10Wunderlinge.txt
```

Mit den Unterschied, dass bei der ersten Variante keine zusätzliche Datei angelegt wird.
Du kannst beliebig viele Befehle mit dem Pipe Symbol verbinden.

Grep

Mithilfe des `grep` Befehls kannst du ein oder mehrere Dokumente nach einem Ausdruck durchsuchen. Standardmäßig gibt der Befehl jede Zeile aus, die auf deine Suche passt.

Gibt alle Zeilen aus, die das Wort „Storm“ enthalten:

```
/Dokumente$   grep "Strom" Wunderlinge.txt
```

Genau wie oben, nur dass Groß-und Kleinschreibung ignoriert wird:

```
/Dokumente$   grep -i "Strom" Wunderlinge.txt
```

Findet exakte Übereinstimmungen des Wortes „Storm“, so etwas wie „Stormy“ wird nicht erkannt:

```
/Dokumente$   grep -w "Strom" Wunderlinge.txt
```

Findet Übereinstimmungen von „Storm“ in jeder Datei des aktuellen Ordners:

```
/Dokumente$   grep "Strom" *
```

Findet Übereinstimmungen im aktuellen Ordner und allen Unterordnern:

```
/Dokumente$   grep -r "Strom" *
```

Für jede Übereinstimmung von „Ben“ wird die entsprechende Zeile und 4 Zeilen danach ausgegeben:

```
/Dokumente$   grep -A 4 "Ben" Wunderling.txt
```

Für jede Übereinstimmung von „Ben“ wird die entsprechende Zeile und 4 Zeilen davor ausgegeben:

```
/Dokumente$   grep -B 4 "Ben" Wunderling.txt
```

Für jede Übereinstimmung von „Ben“ wird die entsprechende Zeile und 4 Zeilen danach und 4 Zeilen davor ausgegeben (9 Zeilen insgesamt):

```
/Dokumente$   grep -A 4 "Ben" Wunderling.txt
```

Hier werden nicht sie Zeilen ausgegeben, sondern die Dateinamen, in denen eine Übereinstimmung gefunden wurde:

```
/Dokument$    grep -l „Pym“ *
```

Zeigt die Zeilennummer zusammen mit der entsprechenden Zeile an:

```
/Dokumente$   grep -n "Pym" Wunderling.txt
```

Cat
Der `cat` Befahl fügt mehrere Dateien zusammen und gibt sie aus, als wären sie eine Datei.

Gebe drei Dateien aus:

```
/Dokumente$   cat Groot.txt Rocket.txt Starlord.txt
```

Denk daran, dass die Ausgabe nur in deinem Terminal erscheint. Um eine neue Datei mit dem Inhalt zu erstellen nutzt du zum Beispiel diesen Befehl:

```
/Dokumente$   cat Groot.txt Rocket.txt Starlord.txt > Hueter.txt
```

Wenn du alle Dateien eines Ordners kombinieren willst, nutzt du den Asterisk (*).

```
/Dokumente$ cat * > AllesKombiniert.txt
```

Head
Der `head` Befehl gibt standardmäßig die ersten 10 Zeilen einer Datei aus:

```
/Dokumente$   head Namen.txt
```

Du kannst auch eine andere Anzahl Zeilen festlegen. Dieser Befehl gibt die erste 15 Zeilen aus:

```
/Dokumente$   head -n 15 Namen.txt
```

Wenn du die ganze Datei ohne die letzten 15 Zeilen ausgeben willst, kannst du eine negative Zahl angeben:

```
/Dokumente    head -n -15 Namen.txt
```

Der `head` Befehl ist sehr nützlich, um einen Blick in eine Datei zu werfen, ohne sie zu öffnen. Dies ist vor allem bei großen Dateien relevant, die lange zum laden brauchen.

Tail
Der `tail` Befehl ist das Gegenstück zum `head` Befehl. Er gibt dir die letzten 10 Zeilen aus:

```
/Dokumente$   tail Namen.txt
```

Oder die letzten 15 Zeilen:

```
/Dokumente$   tail -n 15 Namen.txt
```

Wenn du das ganze Dokument ohne die ersten 15 Zeilen ausgeben willst, kannst du ein Pluszeichen hinzufügen:

```
/Dokumente$   -n +15 Namen.txt
```

Sonstiges
Wie viele Zeilen hat Namen.txt?

```
/Dokumente$   wc -l Namen.txt
```

Reguläre Ausdrücke

Wenn du mit Befehlen wie `grep` arbeitest, kannst du nach Ausdrücken aus Buchstaben, Zahlen und Leerzeichen suchen. Suchst du aber nach einem bestimmten Muster, dann können dir sogenannte **reguläre Ausdrücke** helfen. Diese nutzen bestimmte Symbole, um Muster wie „jeder beliebige Buchstabe“, „jede beliebige Zahl“, „A oder B“, „mindestens ein Kleinbuchstabe“, usw. darzustellen.
An dieser Stelle betrachten wir jedoch nur einen der Operatoren. Der Punkt (.) bedeutet in regulären Ausdrücken „ein beliebiges Zeichen“. Du kannst zum Beispiel nach so etwas suchen:

```
/Dokumente$   grep -i „w.rt“ Woerterbuch.txt
```

Diese Muster passt auf Wörter wie Wort, Wirt, Wert, usw.

Noch ein Beispiel:

```
/Dokumente$   grep -i „n.m.“ Woerterbuch.txt
```

Dieses passt auf „Name“, aber auch auf die Mitte von „ein Muster“ (n Mu), denn „ein beliebiges Zeichen“ kann auch ein Leerzeichen sein.

Viele Wege führen nach Rom

Oft gibt es mehr als eine Möglichkeit, um das gewünschte Ziel zu erreichen.

Beispiel:

```
/Dokumente$   head -n 12 Wunderlinge.txt | tail -n 5
(Gibt die ersten 12 Zeilen von „Wunderlinge.txt“ und davon die letzten 5 Zeilen aus)
```

Ist das gleiche wie:

```
/Dokumente$   tail -n +7 Wunderlinge.txt | head -n 5
(Gibt alles außer die erste 7 Zeilen von „Wunderlinge.txt“ aus und davon die ersten 5
Zeilen)
```

Das gleiche Ergebnis erhältst du durch:

```
/Dokumente$   tail -n +7 Wunderlinge.txt > Zwischenspeicher.txt
/Dokumente$   head -n 5 Ywischenspeicher.txt
/Dokumente$   rm Zwischenspeicher.txt
(Speichert alles außer den ersten 7 Zeilen in der Datei „Zwischenspeicher.txt“ und gibt
davon die ersten 5 Zeilen aus. Anschließend wird „Zwischenspeicher.txt“ gelö scht.)
```