Ein- und Ausgabeumlenkung

- Viele Prozesse lesen und/oder schreiben Daten
  (z.B. `who`, `date`, `cat`, `rm -i`, ...)

- Zu jedem Kommando gibt es voreingestellte *Dateien*, von denen Daten gelesen und/oder in die Daten geschrieben werden:

  – Standard-Eingabe (`stdin`) (meist Tastatur)
  – Standard-Ausgabe (`stdout`) (meist Terminal)
  – Standard-Fehlerausgabe `stderr`) (meist Terminal)

  Beispiel: `cat` (ohne Argument)

- Die Ein-, Aus- und Fehlerausgaben können mit Hilfe der Shell umgelenkt werden: *Ersetzen der voreingestellten Dateien durch andere Dateien*

- **Ausgabeumlenkung**: *Kommando* > *Zieldatei*

  ```
  cat quelle > ziel
  cat > neu
  ```

  Achtung: Überschreiben der Dateien, falls bereits vorhanden

- Anfügen von Ausgabedaten an vorhandene Dateien mit `>>` statt `>`

  ```
  cat quelle >> ziel
  ```

- Ist die Shellvariable `noclobber` gesetzt, so können mit `>` keine Dateien überschrieben werden.

  `>!` erzwingt die Umlenkung

- **Eingabeumlenkung**: *Kommando* < *Quelldatei*

  ```
  rm -i datei < antwort
  ```

- Zusammenfassung von **Fehler- und Ausgabekanal**: *Kommando* >& *Zieldatei*
- getrennte Umlenkung: (*Kommando* > *Ausgabedatei*) >& *Fehlerdatei*

  Kommandos in runden Klammern werden in einer Subshell ausgeführt!
- Eine **Pipeline** leitet die Ausgaben eines Prozesses in den Eingabekanal eines anschließend auszuführenden Prozesses: *Kommando_1* | *Kommando_2*

  ```
  ls -l /bin | more
  ```
- **einfache Hintereinanderschaltung:** *Kommando_1* ; *Kommando_2*
- |& zur zusammengefassten Lenkung von `stdout` und `stderr` des ersten Kommandos in `stdin` des zweiten Kommandos

  ```
  ls -l /bin /gibtsnicht | more
  ls -l /bin /gibtsnicht |& more
  ```

Kommandos zur Mustersuche

`find` *`Verzeichnis Suchkriterien`*	sucht Dateien in *Verzeichnis* und seinen Unterverzeichnissen, die die `Suchkriterien` erfüllen $\longrightarrow$ **Suche nach Dateien**
`grep` *`Muster Datei(liste)`*	sucht Zeilen in *Datei(liste)*, die das *Muster* enthalten $\longrightarrow$ **Suche in Dateien**

Suchkriterien bei `find` (Auswahl)

`-name` *Muster*	Dateien mit Namen *Muster*
`-user` *Muster*	Dateien, die dem Benutzer *Muster* gehören
`-group` *Muster*	Dateien, die der Gruppe *Muster* gehören
`-perm` n	Dateien mit Zugriffsrechten, die dem Wert n entsprechen
`-links` n	Dateien, die n Hard-Links haben
`-mtime` n	Dateien, deren Inhalt vor n Tagen verändert wurde
`-type` t	Dateien vom Typ t ($t \in \{f, d, l, b, c, p\}$)
`-local`	Dateien auf dem lokalen System
`-mount`	Dateien auf dem Dateisystem, auf dem die Suche beginnt
`-print`	Ausgabe des Pfadnamens der gefundenen Dateien
`-exec` *cmd* {} \;	Ausführung des Kommandos *cmd* mit den gefundenen Dateien als Argument
!	negiert das Suchkriterium
$+/-n$ statt n	mehr/weniger als n

Die Argumente von `grep`

- Es werden alle Zeilen der Dateien auf `stdout` ausgegeben, die das Suchmuster (erstes Argument) enthalten.
- wichtige Optionen:
  - `-n` Zeilennummer als Präfix mit ausgeben
  - `-l` nur die Dateien aus der Dateiliste ausgeben, die das Suchmuster enthalten
  - `-s` unterdrücke Fehlermeldungen
  - `-i` *ignore case distinction*
- `grep`-Aufruf mit nur einem Argument: Durchsuchen von `stdin`

```
ls -l /usr/bin | grep uucp
```

- erstes Argument (Suchmuster): **(UNIX-) regulärer Ausdruck**

Reguläre Ausdrücke in UNIX

Ausdruck	*Bedeutung*	`ed, grep`	`egrep, more`	`vi`
Zeichen	dieses Zeichen	✓	✓	✓
`.`	ein beliebeiges Zeichen	✓	✓	✓
`[...]`	Zeichenklasse	✓	✓	✓
`[^...]`	negierte Zeichenklasse	✓	✓	✓
`*`	beliebige Wiederholung des letzten Ausdrucks	✓	✓	✓
`+`	Wdh.: mind. ein Vorkommen		✓	
`?`	Wdh.: 0 bis 1 Vorkommen		✓	
`\{`n`\}`	Wdh.: n Vorkommen	✓		✓
`\{`n`,\}`	Wdh.: mind. n Vorkommen	✓		✓
`\{`n`,`m`\}`	Wdh.: n bis m Vorkommen	✓		✓

Reguläre Ausdrücke in UNIX (2)

Ausdruck	*Bedeutung*	`ed, grep`	`egrep, more`	`vi`
`^`	Vorkommen am Zeilenanfang	✓	✓	✓
`$`	Vorkommen am Zeilenende	✓	✓	✓
`|`	Oder-Verknüpfung		✓	
`(...)`	Gruppierung		✓	
`\(...\)`	Haltespeicher-Definition	✓		✓
`\`n	Aufruf n-ter Haltespeicher	✓		✓

Beispiele:

`.*` $\leadsto$ beliebige Zeichenkette

`\([a-z]*\)\1` $\leadsto$ Wort der Form ww

- Zeichen mit Sonderbedeutung in der Shell müssen apostrophiert werden!