1.9.F Anwendung/Kommandosubstitution – Ergänzungen und Bilder

1.9.F.1 Ergänzende Bilder zu Kommandosubstitution und `xargs`

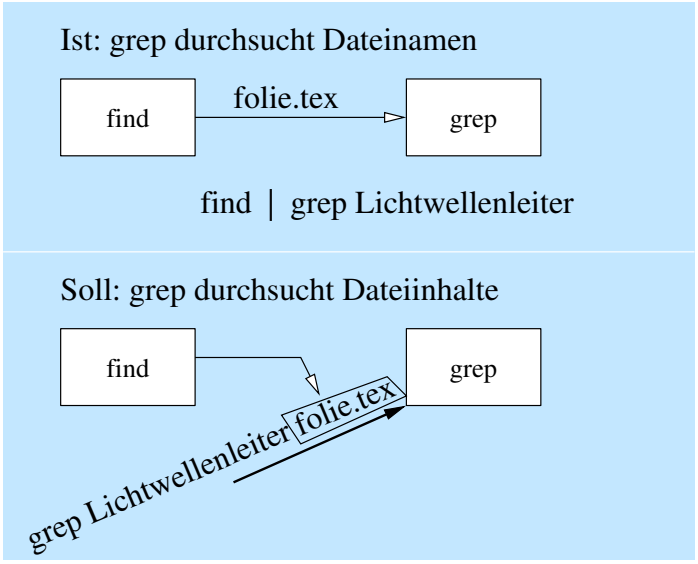

Abbildung 1: Von der Befehlsverkettung (IST) zur Kommandosubstitution (SOLL)

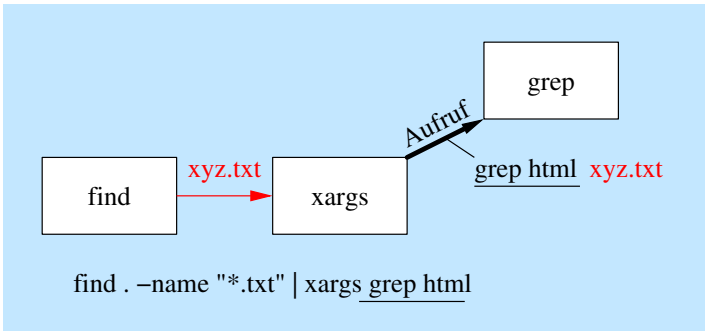

Abbildung 2: Eine typische Anwendung für `xargs`

1.9.F.2 Verschachtelte Kommandosubstitution mit Backticks

Auch bei der Kommandosubstitution in der Schreibweise mit Backticks ist Verschachtelung möglich. Bei der inneren (=zuerst ausgeführten) Kommandosubstitution werden die Backticks durch je einen Backslash maskiert:

Terminal

```
schueler@debian964:~$ echo -e "Kommendes Jahr "
schueler@debian964:~$ echo "ist $(expr "$(date "+%Y")" + 1)."
Kommendes Jahr ist 2025.
schueler@debian964:~$ echo -e "Kommendes Jahr "
schueler@debian964:~$ echo "ist `expr \`date +%Y\` + 1`."
Kommendes Jahr ist 2025.
```

Die Anzahl der Backslashes ist in Tabelle 1 aufgeführt.

Ebene	1	2	3	4	n
Anzahl	0	1	3	7	$2^{n-1}-1$

Tabelle 1: Anzahl der Backslashes bei verschachtelter Kommandosubstitution mit Backticks

1.9.F.3 `xargs` und `read`

Leider ist `xargs` nicht für das Einlesen einer Variable benutzbar:

Terminal
```
schueler@debian964:~$ DATUM=$(date) # funktioniert
schueler@debian964:~$ date | xargs DATUM= # funktioniert nicht
schueler@debian964:~$ date | read DATUM # funktioniert auch nicht
```

Gründe:

- Pipeline-Ausdrücke werden in einer eigenen Subshell ausgeführt. Nach dem Ende der Subshell sind alle Variablen vergessen.
- `DATUM=` ist ein eingebauter Shell-Befehl. `xargs` kann nur externe Programme aufrufen, keine eingebauten Befehle, Aliase und Funktionen.