

Vorlesung Linux-Praktikum

2. Ausgabeumleitung und weitere Befehle

Dirk Frettlöh

Technische Fakultät
Universität Bielefeld

Zusammenfassung heute

bc	einfacher Taschenrechner
>, >>, <	Aus-/Eingabe umleiten
sort	Sortieren
;	mehrere Befehle in einer Zeile trennen
	“Pipe”, Ausgabe des letzten Befehls als Eingabe des nächsten Befehls nehmen
cat	Aneinanderhängen
Strg-c	Programm abbrechen
echo	Argument ausgeben
diff	Unterschied zwischen zwei Dateien anzeigen
head, tail	Anfang/Ende einer Datei anzeigen
grep	nach Zeichenkette in Datei suchen
sed	Suchen und Ersetzen von Zeichenketten

Ein-/Ausgabeumleitung

bc (basic calculator)

- ▶ ein Kommandozeilen - Taschenrechner

```
$ bc
```

```
4 + 7
```

```
11
```

```
9 * 3
```

```
27
```

```
quit
```

Ein-/Ausgabeumleitung

Grundidee



Ein- und Ausgabe sind hier **Text**

Der Standard ist:

- ▶ `stdin`: (Standard Input) Tastatur
- ▶ `stdout`: (Standard Output) Monitor
- ▶ `stderr`: (Standard Error, Fehlermeldung) Monitor

Ein-/Ausgabeumleitung

Grundidee



Ein- und Ausgabe sind hier **Text**

Der Standard ist:

- ▶ `stdin`: (Standard Input) Tastatur
- ▶ `stdout`: (Standard Output) Monitor
- ▶ `stderr`: (Standard Error, Fehlermeldung) Monitor

Man kann Tastatur und Monitor durch **Textdateien** ersetzen

Ein-/Ausgabeumleitung

Beispiel: Ausgabeumleitung

Der Befehl `echo` gibt einfach seine Argumente an die Ausgabe.

```
$ echo Hallo Welt
```

```
Hallo Welt
```

Ein-/Ausgabeumleitung

Beispiel: Ausgabeumleitung

Der Befehl `echo` gibt einfach seine Argumente an die Ausgabe.

```
$ echo Hallo Welt  
Hallo Welt
```

Also:

```
$ echo Hallo Welt > ausgabe.txt
```

Ein-/Ausgabeumleitung

Beispiel: Ausgabeumleitung

Der Befehl `echo` gibt einfach seine Argumente an die Ausgabe.

```
$ echo Hallo Welt  
Hallo Welt
```

Also:

```
$ echo Hallo Welt > ausgabe.txt  
$ more Hallo Welt  
Hallo Welt
```


Ein-/Ausgabeumleitung

Beispiel: Ausgabeumleitung

Der Befehl `echo` gibt einfach seine Argumente an die Ausgabe.

```
$ echo Hallo Welt  
Hallo Welt
```

Also:

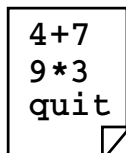
```
$ echo Hallo Welt > ausgabe.txt  
$ more Hallo Welt  
Hallo Welt
```

Falls in `ausgabe.txt` schon etwas steht:


```
> löscht alten Inhalt  
>> hängt an alten Inhalt an
```

Ein-/Ausgabeumleitung

Beispiel: Eingabeumleitung



```
4+7
9*3
quit
```



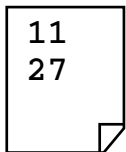
```
$ bc < eingabe.txt
11
27
```

Zeichen für Eingabeumleitung!



Ein-/Ausgabeumleitung

Beispiel: Ausgabeumleitung

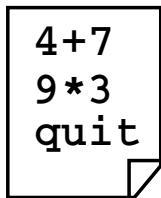


```
$ bc > ausgabe.txt  
4+7  
9*3  
quit
```

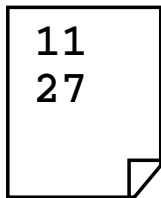
Zeichen für Ausgabeumleitung!

Ein-/Ausgabeumleitung

Beispiel: beides gleichzeitig

A rectangular box with a folded bottom-right corner, representing an input file. It contains the text:

```
4+7  
9*3  
quit
```

A rectangular box with a folded bottom-right corner, representing an output file. It contains the text:

```
11  
27
```

\$ bc < eingabe.txt > ausgabe.txt

Ein-/Ausgabeumleitung

Textdateien zeilenweise sortieren

sort

```
$ sort planeten.txt
```

- ▶ sortiert alphabetisch nach der ersten Spalte

```
$ sort -k 2 planeten.txt
```

- ▶ sortiert alphabetisch nach der **zweiten Spalte**

Ein-/Ausgabeumleitung

Textdateien zeilenweise sortieren

sort

```
$ sort planeten.txt
```

- ▶ sortiert alphabetisch nach der ersten Spalte

```
$ sort -k 2 planeten.txt
```

- ▶ sortiert alphabetisch nach der **zweiten Spalte**

```
$ sort -k 2 -n planeten.txt
```

- ▶ sortiert **numerisch** nach der zweiten Spalte

Ein-/Ausgabeumleitung

Verzeichnislisting nach Größe sortieren

```
$ ls -l > zwischen.txt
```

```
$ sort -k 5 -n zwischen.txt > sort.txt
```

```
$ more sort.txt
```

```
$ rm zwischen.txt sort.txt
```

- ▶ das Hantieren mit temporären Dateien ist lästig!

Ein-/Ausgabeweiterleitung

Ein-/Ausgabeweiterleitung

Grundidee: Verketteten von Programmen



Eingabe →

Programm 1



Programm 2



...



Programm n

→ **Ausgabe**



Ein-/Ausgabeweiterleitung

Anwendung auf das Sortierproblem

Das “Pipe”-Symbol | verbindet die Programme:

```
$ ls -l | sort -k 5 -n | more
```

- Ausgabe des links von | stehenden Programms
- wird Eingabe des rechts von | stehenden Programms
- ▶ deutlich effizienter als Zwischenspeichern!

Kommandos zum Bearbeiten von Textdateien

Textdateien zusammenfügen

cat (concatenate files)

```
$ cat eins.txt zwei.txt drei.txt
```

- ▶ gibt den Inhalt der Dateien nacheinander aus.

```
$ cat eins.txt zwei.txt drei.txt > sammlung.txt
```

- ▶ Ergebnis in neuer Datei speichern.

```
$ cat eins.txt
```

- ▶ Nützlicher Spezialfall: Eine kurze Datei anschauen

Kommandos zum Bearbeiten von Textdateien

Textdateien zusammenfügen

cat (concatenate files)

```
$ cat eins.txt zwei.txt drei.txt
```

- ▶ gibt den Inhalt der Dateien nacheinander aus.

```
$ cat eins.txt zwei.txt drei.txt > sammlung.txt
```

- ▶ Ergebnis in neuer Datei speichern.

```
$ cat eins.txt
```

- ▶ Nützlicher Spezialfall: Eine kurze Datei anschauen

cat ohne Argument liest von der Standardeingabe.

Nützlich: Strg-c bricht Programm ab.

Kommandos zum Bearbeiten von Textdateien

Textdateien zeilenweise vergleichen

diff (show difference between files)

```
$ diff links.txt rechts.txt
```

Entziffern der Ausgabe von diff:

n**c****m**: Die nachfolgenden Zeilen wurden verändert.

<: ursprünglicher Text war in Zeile *n* in links.txt

>: veränderter Text ist in Zeile *m* in rechts.txt

```
8c9
```

```
< ac turpis egestas. In imperdiet porta elit.
```

```
— — —
```

```
> ac turpis egestas. In imperdiet magna elit.
```

Kommandos zum Bearbeiten von Textdateien

Textdateien zeilenweise vergleichen (Forts.)

nam: in der Datei *rechts.txt* **hinzugefügte** Zeilen

3a4

```
> nisi vulputate euismod sollicitudin, dolor quis
```

ndm: in der Datei *rechts.txt* **gelöschte** Zeilen

19d19

```
< fringilla facilisis nisi. Proin id lorem a ipsum
```

Kommandos zum Bearbeiten von Textdateien

Anfang einer Datei ausgeben

head (show head of file)

```
$ head -3 liste.txt
```

- ▶ zeigt die ersten 3 Zeilen einer Datei.

Kommandos zum Bearbeiten von Textdateien

Ende einer Datei ausgeben

tail (show tail of file)

```
$ tail -4 liste.txt
```

- ▶ zeigt die letzten 4 Zeilen einer Datei.

```
$ tail +7 liste.txt
```

- ▶ zeigt alle Zeilen ab der 7ten Zeile
(bzw. unterdrückt die Zeilen 1 bis 6)

Kommandos zum Bearbeiten von Textdateien

Zusammenfassendes komplexes Beispiel

Aufgabe: Planeten-Tabelle mit Überschrift sortieren

```
$ sort planeten2.txt
```

- ▶ klappt nicht wegen der Überschrift

Ansatz: Überschrift mit tail abschneiden

```
$ tail -n +3 planeten2.txt | sort
```

- ▶ besser, aber Überschrift fehlt jetzt

Kommandos zum Bearbeiten von Textdateien

Zusammenfassendes komplexes Beispiel

Überschrift erhält man mit head:

```
$ head -2 planeten2.txt
```

Alles zusammenfügen:

```
$ head -2 planeten2.txt > teil1.txt
```

```
$ tail -n +3 planeten2.txt | sort > teil2.txt
```

```
$ cat teil1.txt teil2.txt > sortiert.txt
```

```
$ rm teil1.txt teil2.txt
```

- ▶ aber es entstehen wieder die unschönen Zwischendateien!

Kommandos zum Bearbeiten von Textdateien

Zusammenfassendes komplexes Beispiel

Es geht auch ohne Zwischendateien:

```
$ head -2 planeten2.txt; tail -n +3 planeten2.txt | sort
```

Semikolon trennt Aufrufe

- ▶ man kann mehr als ein Programm pro Zeile ausführen
- ▶ Ausführung von links nach rechts
- ▶ Ausgaben werden aneinandergehängt

Kommandos zum Bearbeiten von Textdateien

Ausgabeumleitung des Ergebnisses

```
$ head -2 planeten2.txt; tail -n +3 planeten2.txt | sort  
                                     > ergebnis.txt
```

- ▶ liefert nicht das Gewünschte:
nur die Ausgabe von `tail` wird umgeleitet

Lösung:

```
$ (head -2 planeten2.txt; tail -n +3 planeten2.txt | sort)  
                                     > ergebnis.txt
```

- ▶ gesamten Ausdruck in runden Klammern ausführen, dessen Ausgabe umgeleitet werden soll

Kommandos zum Bearbeiten von Textdateien

Texte in Dateien suchen: `grep`

`grep` (global regular expression print)

```
$ grep Mars planeten.txt  
planeten.txt: Mars 6.749 210
```

Durchsucht die Datei `planeten.txt`,
ob sie den Text "Mars" enthält.

Man kann mit Wildcards natürlich mehrere Dateien durchsuchen:

```
$ grep Mars *.txt  
planeten.txt: Mars 6.749 210  
planeten2.txt: Mars 6.749 210
```

Kommandos zum Bearbeiten von Textdateien

Texte in Dateien suchen: `grep`

```
$ grep mars planeten.txt
```

findet keinen Treffer: `mars` \neq `Mars`

Falls Groß-/Kleinschreibung (Datei/datei) egal sein soll:

```
grep -i mars planeten.txt
```

```
planeten.txt:4 Mars 6.749 210
```

```
...
```

Kommandos zum Bearbeiten von Textdateien

Ausgaben mit grep filtern

Filtern von Programmausgaben mit grep:

```
ls -la | grep 2024
```

- ▶ zeigt alle Dateien mit Datum 2024

Kommandos zum Bearbeiten von Textdateien

Ausgaben mit grep filtern

Filtern von Programmausgaben mit grep:

```
ls -la | grep 2024
```

- ▶ zeigt alle Dateien mit Datum 2024

```
ls -la | grep :
```

- ▶ zeigt alle Dateien jünger als ein Jahr
(wegen des speziellen Formats von `ls -l`, ansehen!)

Suchen und Ersetzen

Bielefled;21243;mittel;Station 44;1.Januar 2020

Herford;5741;hoch;Mast 38;1.Januar 2020

Gütersloh;28759;mittel;Mast 92;1.Januar 2020

Bielefled;12535;hoch;Mast 81;2.Januar 2020

Herford;20885;niedrig;Mast 3;2.Januar 2020

...

- ▶ wir brauchen "Suchen und Ersetzen" für die Kommandozeile

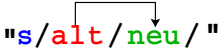
Suchen und Ersetzen

innerhalb von Textdateien

sed: script editor - "Suchen und Ersetzen" per Kommandozeile

Ersetzen des ersten Vorkommens:

```
$ echo "alt alt alt" | sed "s/alt/neu/"  
$ neu alt alt
```



Ersetzen aller Vorkommen:

```
$ echo "alt alt alt" | sed "s/alt/neu/g"  
$ neu neu neu
```

- ▶ **s** - Betriebsart (hier: Ausdruck suchen und ersetzen; es gibt noch weitere, aber s ist die häufigste)
- ▶ **g** - Modifier (hier: globale Ersetzung)

Suchen und Ersetzen

Fehler in der Tabelle korrigieren

```
$ sed "s/Bielefled/Bielefeld/g" < messung-typo.csv
```

```
Bielefeld;21243;mittel;Station 44;1.Januar 2020
```

```
Herford;5741;hoch;Mast 38;1.Januar 2020
```


```
Gütersloh;28759;mittel;Mast 92;1.Januar 2020
```

```
Bielefeld;12535;hoch;Mast 81;2.Januar 2020
```

```
...
```

Andere Betriebsart für sed

```
> echo "alt alt alt" | sed -e "s/alt/neu/"  
> neu alt alt
```



Betriebsart `y`:

Buchstaben aus **Liste1** durch diejenigen aus **Liste2** ersetzen

Suchen und Ersetzen

Beispiel

(den folgenden Ausdruck in eine Zeile schreiben!)

```
$ echo "HALLO" |  
  sed -e "y/ABCDEFGHIJKLMNOPQRSTUVWXYZ  
          /abcdefghijklmnopqrstuvwxyz/"  
hallo
```

Arbeiten mit Tabellen

CSV-Tabellen

CSV: character separated values

Typische Darstellung von Tabellen als Textdateien:

```
Bielefeld;21243;mittel;Station 44;1.Januar 2021
Herford;5741;hoch;Mast 38;1.Januar 2021
Gütersloh;28759;mittel;Mast 92;1.Januar 2021
Bielefeld;12535;hoch;Mast 81;2.Januar 2021
```

Trennzeichen (hier: ;) beliebig wählbar
solange es nicht innerhalb der Daten vorkommt!

Arbeiten mit Tabellen

Spalten aus CSV-Tabellen auswählen

cut: Spalten aus Tabellen auswählen

Aufruf: `cut -d trennzeichen -f spalten`

Trennzeichen mit Bedeutung in der Shell “entschärfen”:

```
cut -d \;
```

```
cut -d \_
```

Arbeiten mit Tabellen

Spalten aus CSV-Tabellen auswählen

cut: Spalten aus Tabellen auswählen

Aufruf: `cut -d trennzeichen -f spalten`

Trennzeichen mit Bedeutung in der Shell “entschärfen”:

```
cut -d \;
```

```
cut -d \_
```

typische Spaltenauswahlen:

```
cut -f 2,5,9 Spalten 2,5,9 auswählen
```

```
cut -f 2-4,7 Spalten 2 bis 4 und 7
```

```
cut -f 5- alle Spalten ab der 5.
```


Arbeiten mit Tabellen

Beispiele

Spalten 1,2 und 4 auswählen:

```
$ cut -d\; -f1,2,4 messung.csv
```

```
Bielefeld;21243;Station 44
```

```
Herford;5741;Mast 38
```

```
Gütersloh;28759;Mast 92
```

Arbeiten mit Tabellen

Beispiele

Spalten 1,2 und 4 auswählen:

```
$ cut -d\; -f1,2,4 messung.csv  
Bielefeld;21243;Station 44  
Herford;5741;Mast 38  
Gütersloh;28759;Mast 92
```

Spalten 1,2 und 5 nur für Bielefeld auswählen:

```
$ grep Bielefeld messung.csv | cut -d\; -f 1-2,5  
Bielefeld;21243;1.Januar.2021  
Bielefeld;12535;2.Januar.2021  
Bielefeld;24817;3.Januar.2021  
...
```

▶ Spalten vertauschen → Übungen

Tabellen mit Leerzeichen als Spaltentrennern

Ausgabe von ls spaltenweise zerlegen

Ziel: In der Ausgabe von `ls -l` Größe und Namen von Dateien (Spalten 5,9) extrahieren.

Tabellen mit Leerzeichen als Spaltentrennern

Ausgabe von ls spaltenweise zerlegen

Ziel: In der Ausgabe von `ls -l` Größe und Namen von Dateien (Spalten 5,9) extrahieren.

Problem: `cut` betrachtet 3 Leerzeichen als 3 leere Spalten!

```
> ls -l
-rwxr--r-- 1 cg cg 612 20. Nov 14:55 gen.bash
-rw-r--r-- 1 cg cg 12447 20. Nov 14:56 messung.csv
```

unterschiedlich viele Leerzeichen

```
$ ls -l | cut -d\ -f 5,9
Nov
12447 messung.csv
```

Tabellen mit Leerzeichen als Spaltentrennern

einzelne Zeichen umwandeln oder zusammenfassen

tr: Zeichen umwandeln oder zusammenfassen

Zeichen komprimieren:

```
$ echo "abxxxbacxxxxxb" | tr -s "x"  
abxbacxb
```

Zeichen umwandeln:

```
$ echo "abxxxbaccxxxxxb" | tr "xc" "yd"  
abyyybaddyyyyyyb
```

Groß-/Kleinschreibung konvertieren:

```
$ echo GROSS | tr [[:upper:]] [[:lower:]]  
gross
```

Tabellen mit Leerzeichen als Spaltentrennern

Lösung zum Auswählen von Spalten aus ls -l

```
ls -l | tr -s " " | cut -d\ -f 5,9
```

```
612 gen.sh
```

```
238 ls-size.sh
```

```
12447 messung.csv
```

```
283 rechner.sh
```

```
4502 verbrauch.txt
```

```
4096 verzeichnis
```

Zusammenfassung heute

bc	einfacher Taschenrechner
>, >>, <	Aus-/Eingabe umleiten
sort	Sortieren
;	mehrere Befehle in einer Zeile trennen
	“Pipe”, Ausgabe des letzten Befehls als Eingabe des nächsten Befehls nehmen
cat	Aneinanderhängen
Strg-c	Programm abbrechen
echo	Argument ausgeben
diff	Unterschied zwischen zwei Dateien anzeigen
head, tail	Anfang/Ende einer Datei anzeigen
grep	nach Zeichenkette in Datei suchen
sed	Suchen und Ersetzen von Zeichenketten
cut	einzelne Einträge aus Zeilen auswählen
tr	<i>trim</i> , stutze Zeichenketten zurecht

Ende der heutigen Vorlesung

Sie probieren das alles gleich in den Tutorien aus.

Viel Spaß! Bis morgen!