Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

The Ninth International Conference on Advances in Databases, Knowledge, and Data Applications

Mai 21 - 25, 2017 - Barcelona, Spain

Data Manipulation and Data Transformation using the Shell

Andreas Schmidt[1,2] and Steffen G. Scholz[2]

(1)
Department of Informatics and
Business Information Systems
University of Applied Sciences Karlsruhe
Germany

(2)
Institute for Applied Computer Sciences
Karlsruhe Institute of Technologie
Germany

Resources available

http://www.smiffy.de/dbkda-2017/ [1]

- Slideset
- Exercises
- Command refcard
- Example datasets

1. all materials copyright, 2017 by andreas schmidt

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Outlook

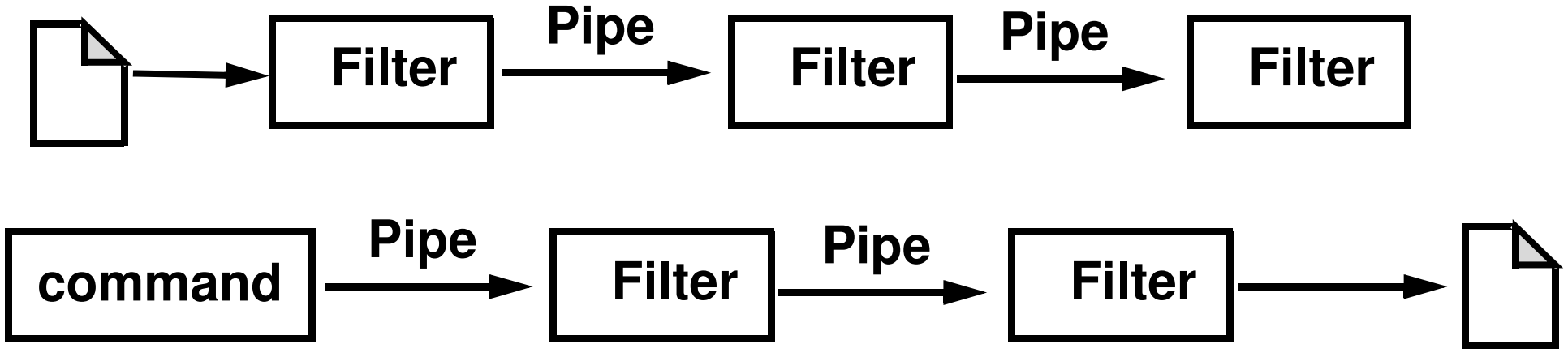
- Overview
- Search and Inspect
- File operations
- Excursus Regular Expressions
- sed & awk
- Emulating SQL with the Shell
- Summary

+ 3 hands on exercices

- First contact
- Analyzing text
- sed & awk

Data Processing with the Shell

- Architectural Pattern: Pipes and Filters (Douglas McIlroy, 1973)
- Data exchange between processes
- Loose coupling
- POSIX Standard
- Filter represent data-sources and data-sinks



Shell commandos in the Linux/Unix/Cygwin Environment

- Input-/Output channels
  - Standardinput (STDIN)
  - Standardouput (STDOUT)
  - Standarderror (STDERR)
- In-/Output Redirection
  - > : Redirect Standardoutput (into file)
  - < : Redirect Standardinput (from file)
  - 2> : Redirect Standarderror (into file)
  - >> : Redirect Standardoutput (append into file)
  - | : Pipe operator: Connect Standardoutput of a command with Standardinput of the next command
- Example:

```
cut -d, -f1 city.csv|sort|uniq -c|sort -nr|awk  '$1>1'>result.txt
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Overview over Operations

- File inspection
- Column/Row extraction
- Filtering
- Searching
- String substitution
- Splitting and Merging files
- Sorting
- Counting
- Insert/Append/Delete lines
- Join-Operations
- Aggregation
- Set Operations
- Compression
- Operations on compressed data

Example File city.csv

```
Aachen,D,"Nordrhein Westfalen",247113,NULL,NULL
Aalborg,DK,Denmark,113865,10,57
Aarau,CH,AG,NULL,NULL,NULL
Aarhus,DK,Denmark,194345,10.1,56.1
Aarri,WAN,Nigeria,111000,NULL,NULL
Aba,WAN,Nigeria,264000,NULL,NULL
Abakan,R,"Rep. of Khakassiya",161000,NULL,NULL
Abancay,PE,Apurimac,NULL,NULL,NULL
Abeokuta,WAN,Nigeria,377000,NULL,NULL
Aberdeen,GB,Grampian,219100,NULL,NULL
Aberystwyth,GB,Ceredigion,NULL,NULL,NULL
Abidjan,CI,"Cote dIvoire",NULL,-3.6,5.3
Abilene,USA,Texas,108476,-99.6833,32.4167
"Abu Dhabi",UAE,"United Arab Emirates",363432,54.36,24.27
Abuja,WAN,Nigeria,NULL,NULL,NULL
Acapulco,MEX,Guerrero,515374,NULL,NULL
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Example File country.csv

```
Austria,A,Vienna,Vienna,83850,8023244
Afghanistan,AFG,Kabul,Afghanistan,647500,22664136
"Antigua and Barbuda",AG,"Saint Johns","Antigua and Barbuda",440,65647
Albania,AL,Tirane,Albania,28750,3249136
Andorra,AND,"Andorra la Vella",Andorra,450,72766
Angola,ANG,Luanda,Luanda,1246700,10342899
Armenia,ARM,Yerevan,Armenia,29800,3463574
Australia,AUS,Canberra,"Australia Capital Territory",7686850,18260863
Azerbaijan,AZ,Baku,Azerbaijan,86600,7676953
Belgium,B,Brussels,Brabant,30510,10170241
Bangladesh,BD,Dhaka,Bangladesh,144000,123062800
Barbados,BDS,Bridgetown,Barbados,430,257030
Benin,BEN,Porto-Novo,Benin,112620,5709529
"Burkina Faso",BF,Ouagadougou,"Burkina Faso",274200,10623323
Bulgaria,BG,Sofia,Bulgaria,110910,8612757
Bhutan,BHT,Thimphu,Bhutan,47000,1822625
```

General comment

- Most of the commands accept the input from file or from STDIN. If no (or not enough) input files are given, it is expected that the input comes from STDIN

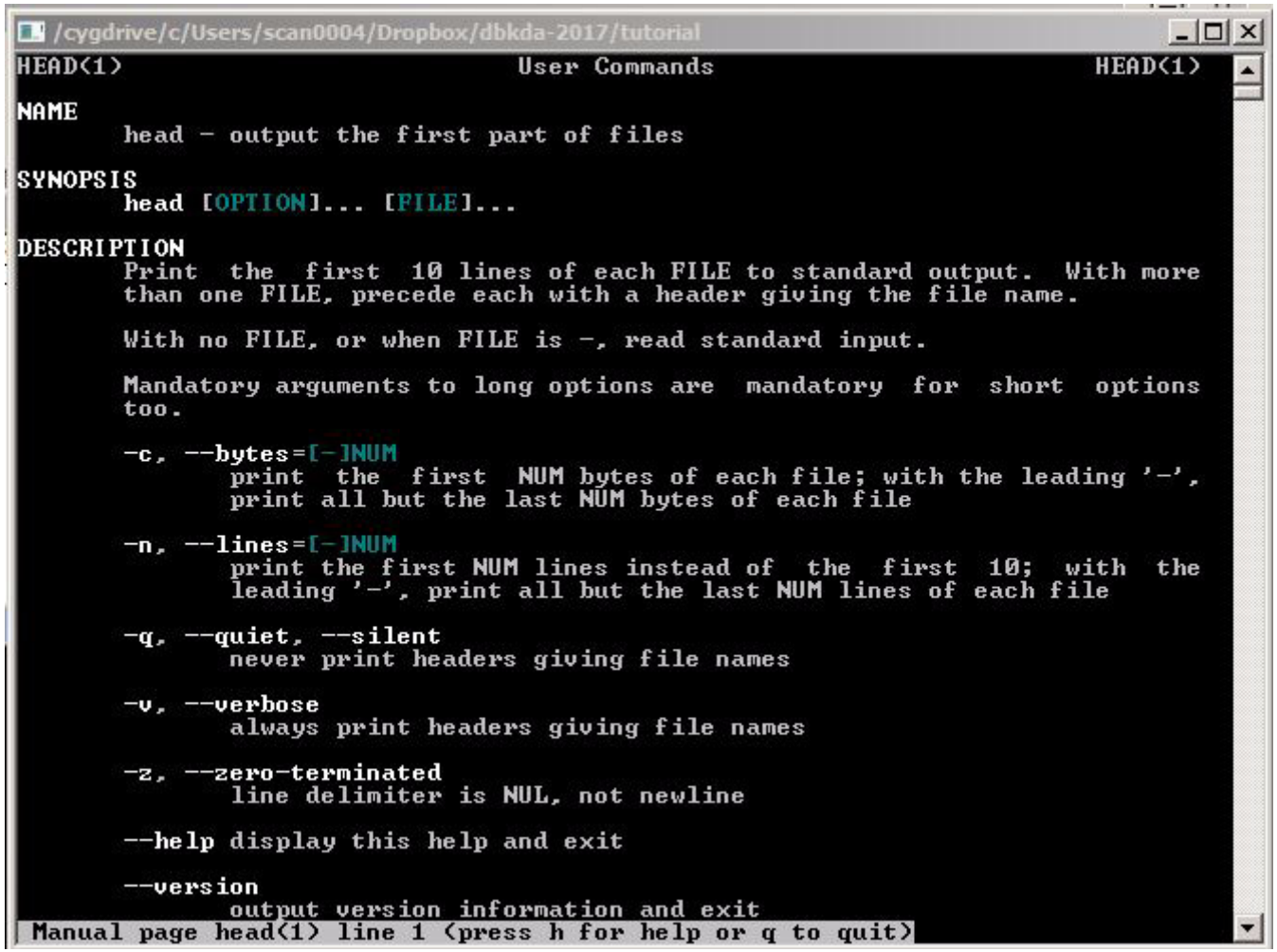
```
head -n4 my-file.txt
cat -n my-file.txt | head -n4
```

- Most of the commands have a lot of options which couldn't be explained in detail. To get an overview of the possibilities of a command, simple type

```
man command
```

- Example:

```
man head
```


/cygdrive/c/Users/scan0004/Dropbox/dbkda-2017/tutorial
HEAD(1) User Commands HEAD(1)
NAME
head - output the first part of files
SYNOPSIS
head [OPTION]... [FILE]...
DESCRIPTION
Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name.
With no FILE, or when FILE is -, read standard input.
Mandatory arguments to long options are mandatory for short options too.
-c, --bytes=[-]NUM
print the first NUM bytes of each file; with the leading '-', print all but the last NUM bytes of each file
-n, --lines=[-]NUM
print the first NUM lines instead of the first 10; with the leading '-', print all but the last NUM lines of each file
-q, --quiet, --silent
never print headers giving file names
-v, --verbose
always print headers giving file names
-z, --zero-terminated
line delimiter is NUL, not newline
--help display this help and exit
--version
output version information and exit
Manual page head(1) line 1 (press h for help or q to quit)

File Inspection

- Show content of a file

```
cat HelloWorld.java
```

- Concatenate files and print them to STDOUT

```
cat german_cities.csv french_cities.csv > cities.csv
cat *_cities.csv > cities.csv
```

- Add line numbers to each line in file(s)

```
cat -n city.csv
```

- Create a file with input from STDIN:

```
cat > grep-strings.txt
Obama
Climate
CTRL-D
```

File Inspection

- View first 5 lines from file:

```
head -n5 city.csv
```

- View last 4 lines of a file with line numbers:

```
cat -n city.csv| tail -n4
```

- View content of file, starting from line 40:

```
tail -n +40 city.csv
```

- Print all but the last 2 lines:

```
head -n -2 city.csv
```

- Count the number of lines, words and bytes

```
wc city.csv
```

- Count the number of lines

```
wc -l city.csv
```

to remove header line(s)

to remove trailing line(s)

less command

- Page by page scrolling of a file or STDIN (also with search capability)
- Examples:

```
less city.csv
ls -l | less

man head        # inspection of man-pages with less !!
```

  - Commands:
    - q : quit less
    - > : Goto end of file
    - < : Goto begin of file
    - f: Scroll forward one page
    - b: scroll backwards on page
    - *e, ret*,↓ : scroll forward one line
    - y,↑: scroll backwards one line
    - *n*d : scroll forward *n* lines (i.e. 20n)
    - *m*b : scroll backwards *m* lines
    - *n*g: Goto line <n>

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

less commands (2)

- /*pattern* : Search forward the next line with *pattern*
- ?*pattern* : Search backward the previous line with *pattern*
- n : repeat previous search
- N : repeat previous search in reverse direction
- &*pattern* : Display only lines containing the *pattern* (type &<*ret*> to quit)
- !*command* : executes shell command
- v : invokes standard editor for file (at current position, if supported)

type **`man less`** for complete reference

Search

- Print lines matching pattern (case sensitive)

```
grep USA city.csv
```

- Print matching lines in a binary file

```
grep -a USA kddNuggests.data
```

- Print lines matching pattern (case insensitive)

```
grep -i town city.csv
```

- Print lines containing the regular expression (City starting with 'S', ending with 'g')

```
grep -E 'S[a-z]+g,' city.csv                    # same as egrep
```

- Print only lines, not containing the String NULL

```
grep -v NULL city.csv
```

- Prefix each line of output with the line number

```
grep -n NULL city.csv
```

Search

- Print all numbers between 1000 and 9999 which have two consequtive 5 in it

```
seq 1000 9999| grep 55
```

- Print only matching part (i.e. 'Salzburg' instead of whole line)

```
grep -E -o 'S[a-z]+g' city.csv
```

- Count the number of times, the word „Karlsruhe“ appears

```
grep -c Karlsruhe famous-cities.txt
```

- Look for lines containing words from file

```
grep -f grep-strings.txt -E newsCorpora.csv
```

  - file: grep-strings.txt

```
Obama
Climate
```

Compression

- gzip compresse files based on LZ77-coding (typ. 60%-70% reduction in size)
- bzip2 compress files based on Huffman coding
- zcat, bzcat, zgrep, bzgrep work on compressed files
- Example:
  - Size:

```
big.txt: 8,9 GB
big.txt.gz: 2.4 GB (gzip -c big.txt > big.txt.gz)
big.txt.bz2: 2.0 GB (bzip2 -c big.txt > big.txt.bz2)
```

  - Runtime:

```
grep something big.txt | wc -l              # ~ 20sec.
zgrep something big.txt.gz | wc -l          # ~ 80 sec.
bzgrep something big.txt.bz2 | wc -l        # ~ 380 sec.
```

Exercise I

- Download the book "The Adventures of Tom Sawyer" from http://www.gutenberg.org/ebooks/74 (utf-8 format).
- For cygwin users: Convert file to Unix Format with command: dos2unix.exe <file>
- Browse (using `less`) through the pages of the book and use some of the commands explained before (page 12)
- Go to line 1234 of the file. What ist the third word?
- How many chapters has the book? (try also the -a option for `grep`)
- Count the number of empty lines
- Execute grep Tom <file> and grep -o Tom <file>. What is the difference?
- How often does the names "Tom" and "Huck" appears in the book?
- How often do they appear together in one line?

File operations

- Split file by row (here, after each 10 lines)

```
split --lines=10 city.csv
```

- Print selected parts of lines from each file to standard output.

```
cut -d',' -f1,4 city.csv
```

Column separator

Output columns

- Output bytes 10 to 20 from each line

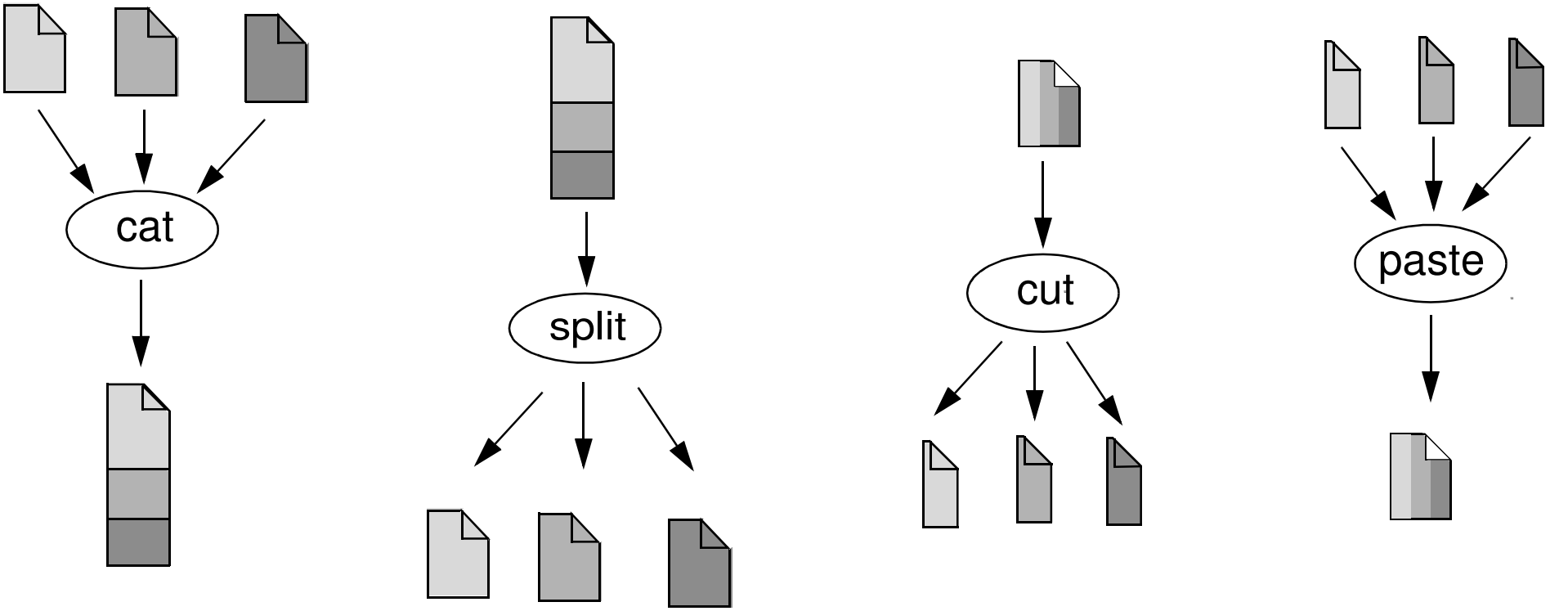
```
cut -b10-20 data.fixed
```

- Merge lines of files

```
paste -d'\t' city_name.txt city_pop.txt > city_name_pop.csv
```

Output delimiter

Summary File operations



tr command

- Translate, squeeze, and/or delete characters from standard input, writing to standard output.
- Translate: Mapping between characters, i.e.
  - {A->a, B->b, ...}
  - {A->*, E->*, I->*, O->*, U->*}
- Delete:
  - {0,1,2,3,4,5,6,7,8,9}
- Squeeze:
  - {aa...a -> a, xx...x -> x, \n\n...\n->\n}
- Predefined character classs/ASCII-Code:
  - [:punct:], [:alnum:], [:alpha:], [:blank:], [:upper:] [:lower:]
  - \xxx : Octal ASCII number (i.e. *<space>* -> \040)

- Examples
  - Translate to lowercase:

```
tr 'A-Z' 'a-z' < The-Adventures-of-Tom-Sawyer.txt
```

  - Replace <newline> with <space>

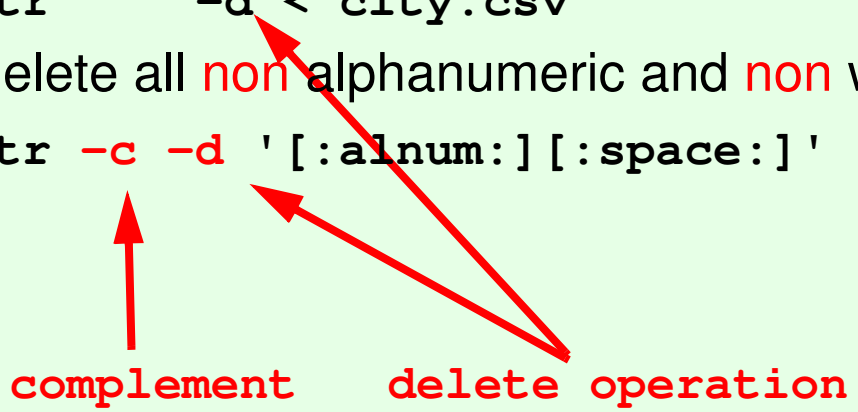
```
tr '\n' '' < short-story.txt > one-liner.txt
```

  - Delete all (") characters

```
tr '"' -d < city.csv
```

  - Delete all non alphanumeric and non whitespace characters

```
tr -c -d '[:alnum:][:space:]' < The-Adventures-of-Tom-Sawyer.txt
```



sort

- Sort lines of text files
- Write sorted concatenation of all FILE(s) to standard output.
- With no FILE, or when FILE is -, read standard input.
- sorting alpabetic, numeric, ascending, descending, case (in)sensitive
- column(s)/bytes to be sorted can be specified
- Random sort option (-R)
- Remove of identical lines (-u)
- Examples:
  - sort file city.csv starting with the second column (field delimiter: ,)

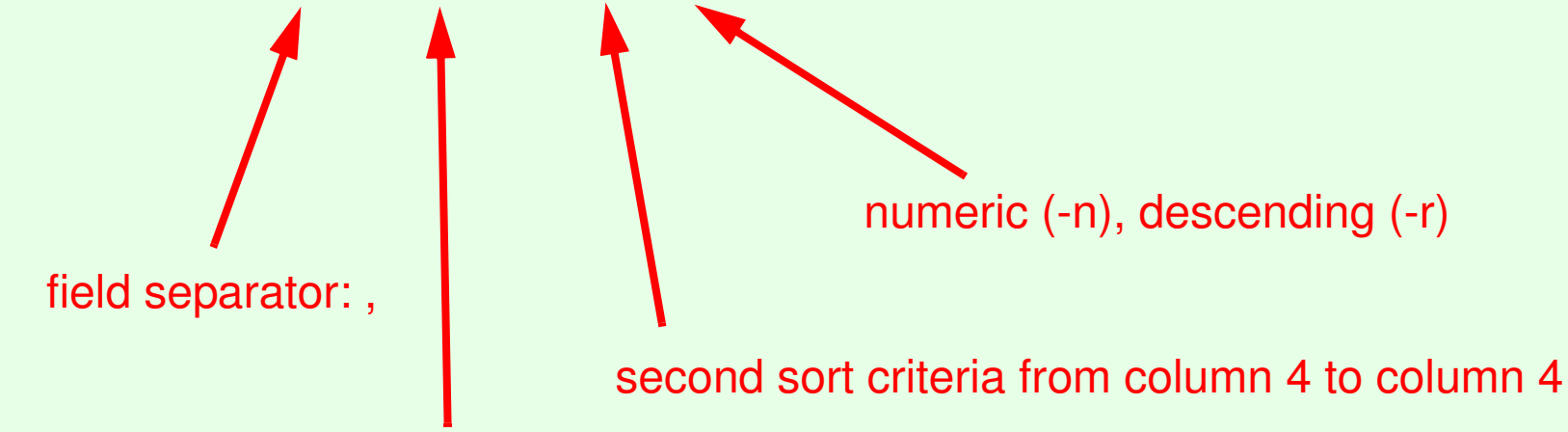
    ```
    sort -k2 -t',' city.csv
    ```
  - merge content of file1.txt and file2.txt and sort the result

    ```
    sort file1.txt file2.txt
    ```

sort - examples

- sort file by country code, and as a second criteria population (numeric, descending)

```
sort -t, -k2,2 -k4,4nr  city.csv
```



sort - examples

- Sort by the second and third character of the first column

```
sort -t, -k1.2,1.2 city.csv
```

- Generate a line of unique random numbers between 1 and 10

```
seq 1 10| sort -R | tr '\n' ' '
```

- Lottery-forecast (6 from 49) - defective from time to time ;-)

```
seq 1 49 | sort -R | head -n6
```

- Test if a file is sorted

```
seq 1 10| sort -R | sort -c
```

Further File operations

- join - join lines of two files on a common field
- Fields to compare must be sorted (alphabetic, not numeric)
- Output fields can be specified
- Example:

```
sort -k2 -t, city.csv | join -t, -12 -22 - country.csv \
                                -o1.1,2.1,1.3,1.4
```

Join Operation

- city.csv

```
Aachen,D,"Nordrhein Westfalen",247113,NULL,NULL
Aalborg,DK,Denmark,113865,10,57
Aarau,CH,AG,NULL,NULL,NULL
Aarhus,DK,Denmark,194345,10.1,56.1
Aarri,WAN,Nigeria,111000,NULL,NULL
...
```

- country.csv

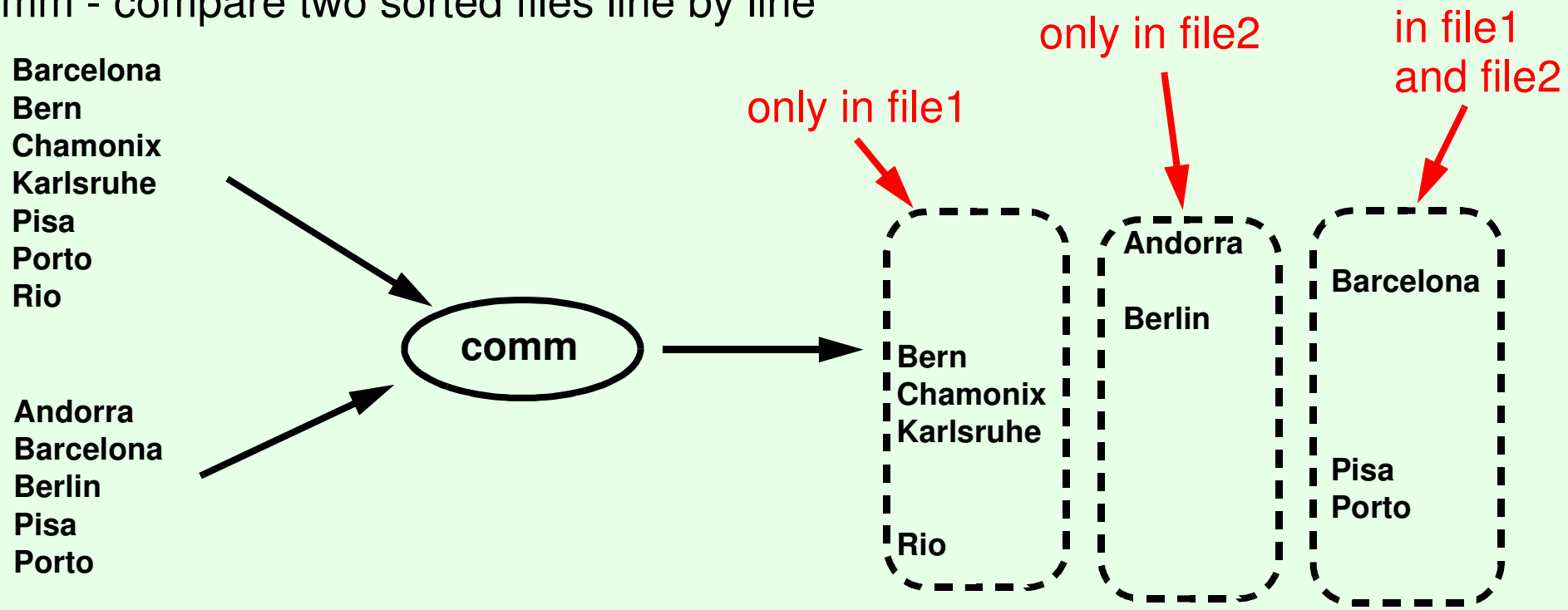
```
...
Germany,D,Berlin,Berlin,356910,83536115
Djibouti,DJI,Djibouti,Djibouti,22000,42764
Denmark,DK,Copenhagen,Denmark,43070,524963
Algeria,DZ,Algiers,Algeria,2381740,2918303
Spain,E,Madrid,Madrid,504750,39181114
...
```

```
sort -k2 -t, city.csv | join -t, -12 -22 - country.csv \
                        -o1.1,2.1,1.3,1.4
```

```
Aachen,Germany,"Nordrhein Westfalen",247113
Aalborg,Denmark,Denmark,113865
Aarau,Switzerland,AG,NULL
Aarhus,Denmark,Denmark,194345
Aarri,Nigeria,Nigeria,111000
Aba,Nigeria,Nigeria,264000
Abakan,Russia,"Rep. of Khakassiya",161000
```

Compare Operator

- comm - compare two sorted files line by line



- Options:
  - -1: supress column 1
  - -2: supress column 2
  - -3: supress column 3
  - --total: output a summary

set semantic with sorted input !!!

uniq

- report or omit repeated lines
- Filter adjacent matching lines from INPUT
- Range of comparision can be specified (first n chars, skip first m chars)
- options:
  - -c: count number of occurences
  - -d: only print duplicate lines
  - -u: only print unique line
  - -i: ignore case

set semantic with sorted input !!!

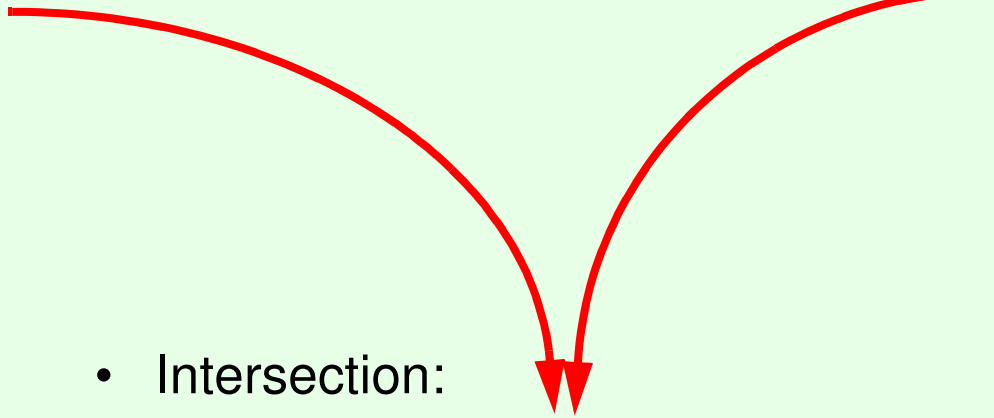
uniq - example

- file1.txt

```
Barcelona
Bern
Chamonix
Karlsruhe
Pisa
Porto
Rio
```

- file2.txt

```
Andorra
Barcelona
Berlin
Pisa
Porto
```



- Intersection:

```
$ cat file*.txt | sort | uniq -d
Barcelona
Pisa
Porto
```

- Counting:

```
cat file*.txt | sort | uniq -c
  1 Andorra
  2 Barcelona
  1 Berlin
  1 Bern
  1 Chamonix
  1 Karlsruhe
  2 Pisa
  2 Porto
  1 Rio
```

Exercise II - (Duration 15 min.)

- Count the words and lines in the book 'The-Adventures-of-Tom-Sawyer.txt'
- What does the following command perform?

```
grep -o -E '[A-Za-z]+' The-Adventures-of-Tom-Sawyer.txt
```

- Translate all words of 'The-Adventures-of-Tom-Sawyer.txt' into lowercase using `tr`
- Count, how often each word in this book appears (hint: use `uniq`)
- Order the result, starting with the word with the highest frequency. Which word is it?
- Write all the above steps in one statement (using pipes)
- Compare the result with the result from the following book: http://www.gutenberg.org/files/2701/2701-0.txt. At which position do the first book specific words appear?
- Compare the 20 most frequent words of each book. How many are in common? (hint: use `head, cut, comm`)

Excursus Regular Expressions

- Character classes:

```
grep '[0-9]' city.csv      # print all lines with a digit in it
grep -v '[0-9]' city.csv   # print all lines without a digit in it
grep '[A-Za-z]' numeric.data # all lines with at least one character
grep '[^AEIOUaeiou]' city.csv # lines with at least one non-vocal
```

- Special characters:
  - [ ] : definition of a character class
  - . : matches any character
  - ^ : matches begin of line or negation inside character class
  - $: matches end of line
  - \b : represents a word boundary

```
grep -a '^The' The-Adventures-of-Tom-Sawyer.txt
grep -a -v '^$' The-Adventures-of-Tom-Sawyer.txt
grep -a -i '\bhouse' The-Adventures-of-Tom-Sawyer.txt
```

Excursus Regular Expressions (2)

- Special characters (continued):
  - | : alternative
  - ( ) : for back referencing

  \n : back reference to (...) (n numeric)

- Examples:

```
egrep ',(USA|TR),' city.csv
egrep  'St\.' city.csv
egrep  'E([a-z])\1' city.csv
```

i.e. St. Louis, but not Stanford

i.e. Essex

If you need to match a . ( ) [ ] { } + \ ^ $ preceed it with a backslash \

Excursus Regular Expressions (3)

- Repetition
  - ? : optional
  - * : zero or more times
  - + : one or more times
  - {n} : exactly *n* times
  - {n,m} : between *n* and *m* times
- Examples:

```
egrep -a -i '\bhouses?\b' The-Adventures-of-Tom-Sawyer.txt
egrep -a 'X{1,3}' The-Adventures-of-Tom-Sawyer.txt
```

matches house, houses

matches X, XX, XXX

String Substitution with sed

- sed - Stream Editor
- non interactiv, controlled by a script
- line oriented text processing
- short scripts are typically given as parameter (-e option),
  longer scripts as files (-f option)
- Possible operations: Insert, Substitude, Delete, Append, Change, Print, Delete
- Commands in script can take an optional *address*, specifying the line(s) to be performed.
- *Address* can be a as ingle line number or a regular expression
- *Address* can be an interval (start, stop)
- A loop executes script commands on each input line
- Default behavior: printing each processed line to stdout (suppress with: -n)

sed commands

- **s**: substitude
  - Replace all occurences of D with GER

```
sed 's/\bD\b/GER/g' city.csv > city2.csv
```

  - Replace „Stuttgart“ with „Stuttgart am Neckar“ (extended regexp)

```
sed -r '/Stuttgart/ s/^([A-Za-z]+)/\1 am Neckar/' city.csv
```

  - Replace all occurences of NULL in a line with \N (Inplace Substitution)

```
sed -i 's/\bNULL\b/\\N/g' city.csv
```

- **p**: print (typically used with default printing behaviour off (-n option))
  - print from line 10 to 20 (resp.: 5-10, 23, 56-71)

```
sed -n 10,20p city.csv
sed -n '5,10p;23p;56,71p' city.csv
```

  - print lines starting from dataset about ’Sapporo’ inclusive dataset about ’Seattle’

```
sed -n '/^Sapporo/,/^Seattle/p' city.csv
```

- **i**: insert
  - Insert dataset about Karlsruhe at line 2

```
sed '2i Karlsruhe,D,"Baden Wuerttemberg",301452,49.0,6.8' city.csv
```

- **d**: delete
  - delete Aachen (inplace)

```
sed -i '/Aachen/ d' city.csv
```

  - delete all empty lines

```
sed '/^ *$/d' The-Adventures-of-Tom-Sawyer.txt
```

  - delete lines 2-10

```
sed '2,10d' city.csv
```

  - delete all <script>..</script> sections in a file

```
sed -Ei '/<script>/,/<\/script>/d' jaccard.html
```

  - delete from `<h2>Navigation menu</h2>` to end of file

```
sed -i '/<h2>Navigation menu<\/h2>/,$d' jaccard.html
```

sed Examples

- **c**: change
  - Replace entry of Biel

```
sed '/^Biel\b/ c Biel,CH,BE,53308,47.8,7.14' city.csv
```

- **a**: append
  - Underline each CHAPTER

```
sed '/^CHAPTER/ a ------------' The-Adventures-of-Tom-Sawyer.txt
```

- ...

awk

- like sed, but with powerful programming language
- filter and report writer
- ideal for processing rows and columns
- suport for associative arrays
- structure: pattern { action statements }
- special BEGIN, END pattern match **before** the first line is read and **after** the last line is read
- Access to column values via $1, $2, ... variables ($0: whole line)
- Examples:

```
awk -F, '$3=="Bayern" && $4 < 1000000 { print $1", "$4 }' city.csv
```

awk

- Calculating average population

```
awk -F, -f average.awk city.csv
```

```
# script: average.awk
BEGIN { sum = 0
        num = 0
}

$4!="NULL" {
  sum += $4
  num++
}

END { print "Average population: "sum/num }
```

pattern

- predefined variables
  - NF: number of fields
  - NR: number of records
  - FS: field separator (default: " ", same as -F from command line)
  - RS: record separator (default: \n)
  - ORS: output record separator
  - FPAT: Field pattern (alternative way to specify a field instead of use of FS)
  - FILENAME

awk example: multi-line input

```
Andreas Schmidt
KIT
Germany

Petre Dini
IARIA
USA

Fritz Laux
University Reutlingen
Germany
```

```
Andreas Schmidt,KIT,Germany
Petre Dini,IARIA,USA
Fritz Laux,University Reutlingen,Germany
```

```
awk -f demo2.awk address.txt
```

```
BEGIN {
    FS="\n"
    RS="\n\n"
    ORS=""
}
{
    for (i=1; i<NF; i++) {
      print $i","
    }
    print $NF"\n"
}
```

awk

- FPAT: Split a line by pattern, rather then by delimiter

- Example:
  - File:

```
12,45,Test, 123.56
13,21,"Test without comma", 345.2
14,71,"Test, with comma", 0.7
```

  - Command:

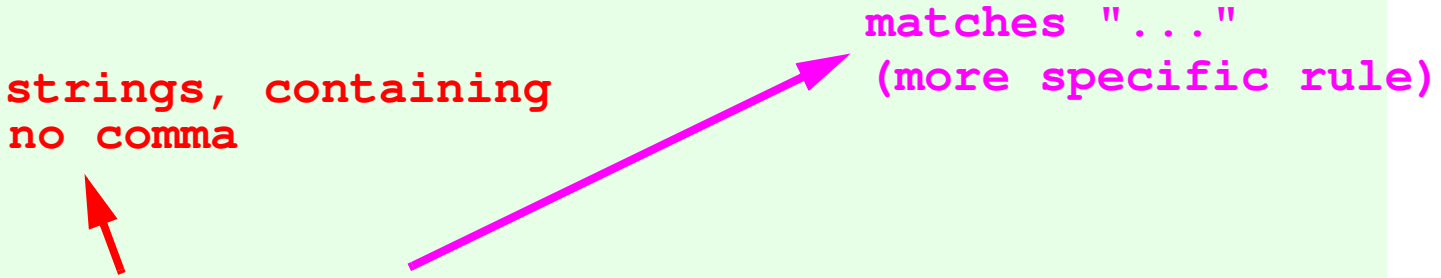
```
awk -F, '{print $1" : "$3}' fpat-demo.txt
```

  - Output:

```
12 : Test
13 : "Test without comma"
14 : "Test
```

WRONG!!!!!

- Example with FPAT:
  - command:




```
awk 'BEGIN{FPAT = "([^,]*)|(\"[^\"]*\")"}   \
     {print $1" : "$3}' fpat-demo.txt
```

  - Output:

```
12 : Test
13 : "Test without comma"
14 : "Test, with comma"
```

Exercise III

- Create a working copy of your file city.csv (for security reasons)
- Exchange all occurences of the Province "Amazonas" in Peru (Code PE) with "Province of Amazonas" using sed (inplace).
- Look for entries with the String "ce of Amazonas" - it should be only 1 !
- Make the same operations using awk.
- Print all cities which have no population given.
- Print the line numbers of the cities in Great Britain (Code: GB)
- Delete the records 5-12 and 31-34 from city.csv and store the result in city2.csv using awk.
- Combine the used commands from the last two tasks and write a bash-script (sequence of commands), which delete all british cities from the file city.csv (Hint: generate with awk the commands for sed to delete the corresponding lines)
- Count the datasets (lines) in city.csv - it should be 2880

Exercise III (continued)

- If you take a look at the files, downloaded from the Gutenberg Project, you can identify some boilerplate text at the begin and the end of the book. Which are the lines, who separate the literary text from the boilerplate text?
- Write a command, which removes the boilerplate text (Hint: use sed, head, tail)

Specifying the field-separator

- Unfortunately, most of the command use a different character to specify the field separator ... and also the default separator differs

Tabelle 1:

command	specification parameter	Default separator
cut	-d	*<Tab>*
sort	-t	*<Blanc>,*
awk	-F	*<Blanc>, <Tab>*
join	-t	*<Blanc>*
uniq		*<Blanc>, <Tab>*

Emulation of SQL-Statements (1)

```
select *                          cat city.csv
  from city

select name, population           cut -d, -f1,4 city.csv
  from city

select name,population            grep ',F', city.csv | cut -d, -f1,4
from city
where country='F'                 awk -F, '$2=="F" {print $1,$4}' city.csv

select count(*)                   wc -l city.csv
from city

select count(*)                   grep ',F', city.csv | cut -d, -f1,4 | wc -l
from city
where country='F'
```

Emulation of SQL-Statements (2)

```
select max(population)                grep ',F,', city.csv | cut -d, -f4 | \
from city                               sort -nr| head -n1
where country='F'

select country, count(*)              cut -d, -f2 city.csv | sort -t,| \
  from city                             uniq -c|sort -nr
 group by country
 order by count(*) desc

select ci.name,                       sort -k2 -t, city.csv | \
       co.name, ci.population         join -t, -12 -22 - country.csv \
  from city ci                             -o1.1,2.1,1.4| \
  join country co                     sort -t, -k3,3 -nr
    on ci.country=co.code
order by ci.population desc
```

Emulation of SQL-Statements (3)

```
select country, count(*)
  from city
 group by country
having count(*) > 100
 order by count(*) desc
```

```
cut -d, -f2 city.csv | sort -t,| \
   uniq -c| sort -nr| \
   awk -F' ' '$1>100 {print}'
```

```
select country, count(*)
  from city c
 where (select area
          from country co
         where co.code=c.country)
          > 5000000
 group by country
having counT(*) > 100
 order by count(*) desc
```

?????

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

update Statement

```
update city
    set population=308135
  where name='Karlsruhe' and country='D';

awk  -F, '{ if ($1=="Karlsruhe" and $2=="D") $4=308135; print $0}' \
city.csv


update city
    set population=round(poulation*1.05)
  where country='D';

awk  -F, '{if ($2=="D") $4=$4*1.05; print $0}' city.csv
```

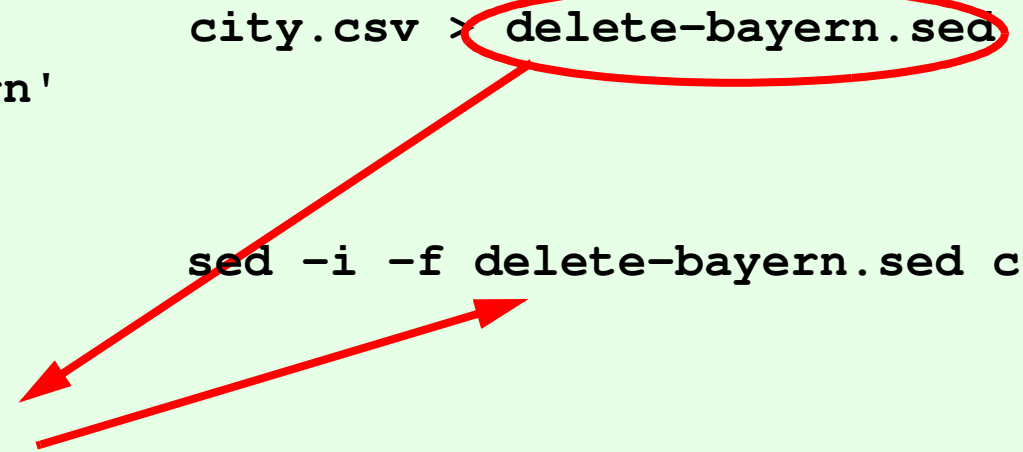
delete statement

```
delete
  from city
 where province='Bayern'
```

```
awk -F, '$3=="Bayern" {print NR"d"}' \
city.csv > delete-bayern.sed
```

```
sed -i -f delete-bayern.sed city.csv
```

```
172d
776d
839d
1094d
1749d
1756d
1904d
2189d
2921d
```



Visualization with Gnuplot

```
awk -F"," ' $4 { print $4, $1 }' city.csv | sort -nr |  head -n30 \
   > biggest-cities.data
```

- biggest-cities.data:

```
10229262 Seoul
9925891 Mumbai
9863000 Karachi
9815795 "Mexico City"
9811776 "Sao Paulo"
8717000 Moscow
8259266 Jakarta
7843000 Tokyo
7830000 Shanghai
```

- Gnuplot file biggest-cities.gpt

```
set terminal postscript
set output "city-population.ps"
set title "City Population"
set style fill solid
set style data histogram
set xtic nomirror rotate by -60
plot "biggest-cities.data" \
     using 1:xtic(2) title ''
```

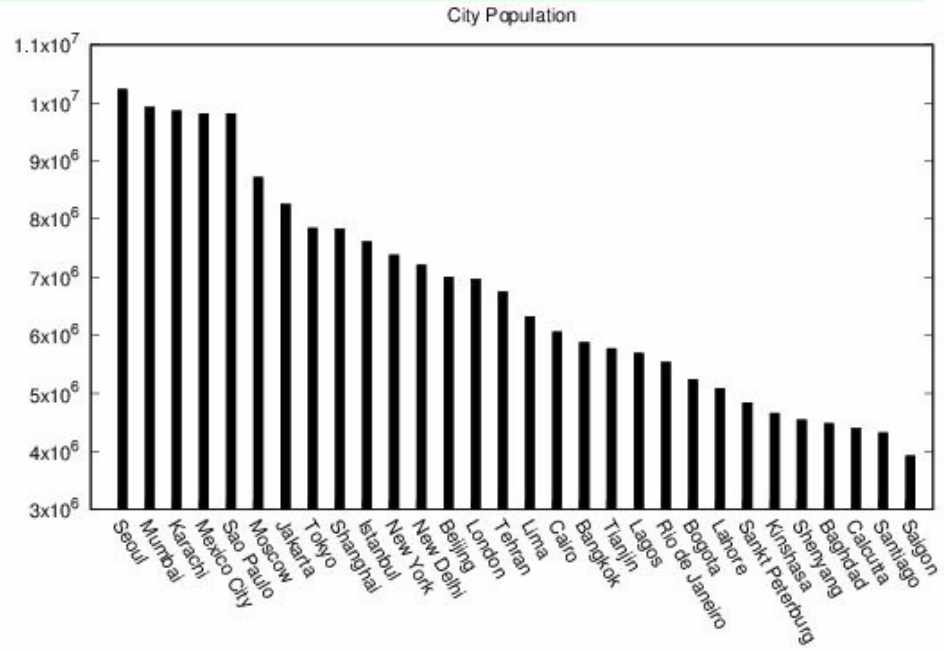
Visualization with Gnuplot

- file biggest-cities.gpt

```
set terminal postscript
set output "city-population.ps"
set title "City Population"
set style fill solid
set style data histogram
set xtic nomirror rotate by -60
plot "biggest-cities.data" \
     using 1:xtic(2) title ''
```

- Generate:

```
$ gnuplot biggest-cities.gpt
```



Visualization with Gnuplot

- file temp.dat

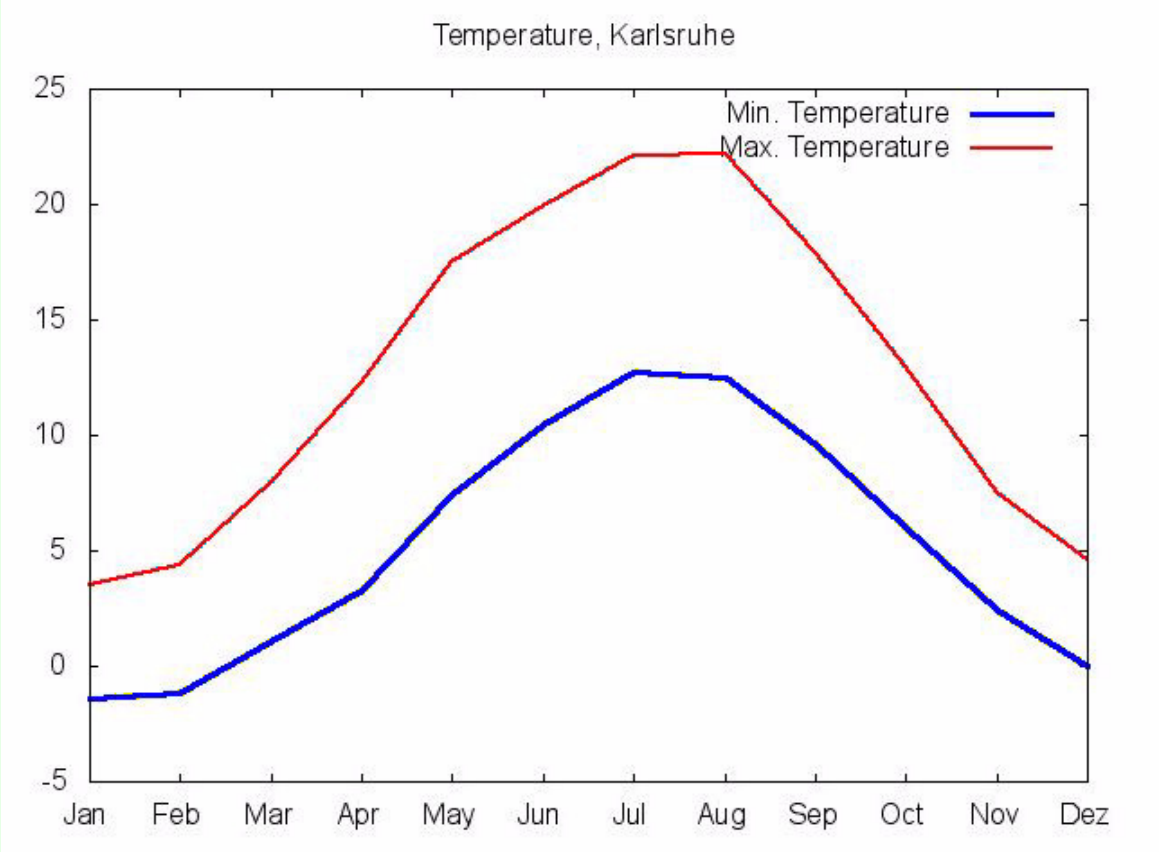
```
Month "Min. Temperature" "Max. Temperature"
Jan   -1.4                3.5
Feb   -1.2                4.4
Mar   1.1                 8.0
Apr   3.3                 12.3
May   7.4                 17.5
Jun   10.5                19.9
Jul   12.7                22.1
Aug   12.5                22.2
Sep   9.6                 17.9
Oct   6.0                 13.0
Nov   2.4                 7.5
Dez   0.0                 4.6
```

- temperature.gpt

```
set term jpeg
set output "temperatur.jpeg"
set title "Temperature, Karlsruhe"
set style line 1 lt 2 lc rgb "blue" lw 3
set style line 2 lt 5 lc rgb "red" lw 2
set multiplot
plot  "temp.dat" using 2:xtic(1) \
                 ls 1 with lines \
                 title columnheader(2), \
      "temp.dat" using 3:xtic(1) \
                 ls 2 with lines \
                 title columnheader(3);
unset multiplot
```

Visualization with Gnuplot

```
$ gnuplot.exe temperature.gpt
```



Further readings

- http://www.theunixschool.com/p/awk-sed.html
- Dale Dougherty, Arnold Robbinssed & awk, 2nd Edition UNIX Power Tools. O'Reilly, 2nd Edition 1997
- Arnold Robbins. Sed and Awk: Pocket Reference, 2nd Edition Paperback – June , O'Reilly, 2002
- Ramesh Natarajan. sed and awk 101 hacks. http://www.thegeekstuff.com/sed-awk-101-hacks-ebook/
- gnuplot homepage: http://www.gnuplot.info/

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Further examples ...

Jaccard Example

```
ENTITY=New_York

wget -O data/$(ENTITY).html https://en.wikipedia.org/wiki/$(ENTITY)
tr < data/$(ENTITY).html > data/$(ENTITY).txt '[A-Z]' '[a-z]'
sed -ri '/<script>/,/<\/script>/d' data/$(ENTITY).txt
sed -ri 's/<!--.*-->/ /g' data/$(ENTITY).txt
sed -ri '/<!--/,/-->/d' data/$(ENTITY).txt
sed -i '/<h2>navigation menu<\/h2>/,$d' data/$(ENTITY).txt
sed -ri 's/<\/?[a-z]+[^>]*>/ /g' data/$(ENTITY).txt
egrep -o -e '[a-z]+' data/$(ENTITY).txt | sort | uniq | \
      $(PHP) porter.php > data/$(ENTITY).set


comm --total data/$(E1).set data/$(E2).set | tail -n1|sed -r 's/([\t0-9]+)[A-
Za-z]+/.\/jaccard.pl \1/g' > run.sh
sh run.sh
```

Create Lookup-Table

```
Innsbruck,Austria,Tyrol,118000
Vienna,Austria,Vienna,1583000
Bregenz,Austria,Vorarlberg,NULL
Kabul,Afghanistan,Afghanistan,892000
"Saint Johns","Antigua and Barbuda",...
Tirane,Albania,Albania,192000
Korce,Albania,Albania,52000
Elbasan,Albania,Albania,53000
Vlore,Albania,Albania,56000
```

```
Innsbruck,1,Tyrol,118000
Vienna,1,Vienna,1583000
Bregenz,1,Vorarlberg,NULL
Kabul,2,Afghanistan,892000
"Saint Johns",3,"Antigua and
Barbuda",36000
Tirane,4,Albania,192000
Korce,4,Albania,52000
Elbasan,4,Albania,53000
Vlore,4,Albania,56000
```

```
1,Austria
2,Afghanistan
3,"Antigua and Barbuda"
4,Albania
5,Andorra
6,Angola
7,Armenia
8,Australia
```

Create Lookup-Table

```
awk -F, -f make_dictionary.awk city_with_countryname.csv
```

```
BEGIN {
    key=0;
    OFS=",";
}
{
  if (has_value[$2]>0)
      out=has_value[$2];
  else {
      has_value[$2] = ++key;
      out=key;
      print key","$2 > "lookup_table.csv"
  }
  $2=out;
  print $0;
}
```

initialisation stuff

reuse existing key

generate next key value

write key/value pair in file

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Looking for doubled paragraphs

```
$ grep -E '.{20,}' diss-tobi.txt| sort | uniq -d > double.grep

$ grep -a -n -f double.grep diss-tobi.txt

3487:Bei einigen Verfahren nivellieren Walzen oder eine Fr¦seinrich-
tung das abgeschiedene
3496:Bei einigen Verfahren nivellieren Walzen oder eine Fr¦seinrich-
tung das abgeschiedene
10352:Design Engineering Technical Conferences & Computers and Infor-
mation in
10529:Design Engineering Technical Conferences & Computers and Infor-
mation in
1079:Diese Ausrichtung bleibt auch, wenn das Feld entfernt wird, und
verleiht dem
4659:Diese Ausrichtung bleibt auch, wenn das Feld entfernt wird, und
verleiht de
```

Simple encryption (like rot13)

```
$ tr 'A-Za-z' 'D-Za-zABC' < top-secret.plain > top-secret.enc

tr 'D-Za-zABC' 'A-Za-z' < top-secret.enc
```