The 13th International Conference on Advances in Databases, Knowledge, and Data Applications

DBKDA 2021
May 30, 2021 to June 3, 2021 - Valencia, Spain

Tutorial: "Bashing" the Killer from the Command Line

Andreas Schmidt¹, Lisa Ehrlinger², and Fritz Laux³

(1) andreas.schmidt@h-ka.de
University of Applied Sciences Karlsruhe
Karlsruhe, Germany &
Karlsruhe Institute of Technology
Germany

(2) lisa.ehrlinger@jku.at
Johannes Kepler University
Linz, Austria &
Software Competence Center
Hagenberg, Austria

(3) fritz.laux@fh-reutlingen.de
Reutlingen University
Reutlingen, Germany
A short Resume of the Presenters

Prof. Dr. Andreas Schmidt is a professor at the Department of Computer Science and Business Information Systems of the Karlsruhe University of Applied Sciences (Germany). He is lecturing in the fields of database information systems, data analytics and model-driven software development. Additionally, he is a senior research fellow in computer science at the Institute for Applied Computer Science of the Karlsruhe Institute of Technology (KIT). His research focuses on database technology, knowledge extraction from unstructured data/text, Big Data, and generative programming. Andreas Schmidt was awarded his diploma in computer science by the University of Karlsruhe in 1995 and his PhD in mechanical engineering in 2000. Dr. Schmidt has numerous publications in the field of database technology and information extraction. He regularly gives tutorials on international conferences in the field of Big Data related topics and model driven software development. Prof. Schmidt followed sabbatical invitations from renowned institutions like the Systems-Group at ETH-Zurich in Switzerland, the Database Group at the Max-Planck-Institute for Informatics in Saarbrücken/Germany and the Data-Management-Lab at the University of Darmstadt.

Dr. Lisa Ehrlinger is senior researcher at the Johannes Kepler University (JKU) Linz and at the Software Competence Center Hagenberg (SCCH) in Austria. At JKU, she conducts research about the automation of DQ measurement and knowledge graphs and teaches courses about information systems, ontologies, and data modeling. At SCCH, she leads the multi-firm project SEBISTA (Secure Big Stream Data Processing), where she applies finding from her scientific work, and drives the research focus “Data Management and Data Quality”. Her research interests and publications cover the topics data quality, knowledge graphs, and information integration.
A short Resume of the Presenters

Prof. Dr. Fritz Laux was professor (now emeritus) for Database and Information Systems at Reutlingen University from 1986 - 2015. He holds an MSc (Diplom) and PhD (Dr.rer. nat.) in Mathematics.

His research focuses on database modeling and technology, transaction processing, data warehousing and data analytics. He has published a number of papers in peer reviewed conferences and journals on the above topics, some of them have received Best Paper Awards. He is a regular contributor and speaker at DBKDA.

Prof. Laux is a co-founder of DBTechNetDB, an initiative of European universities and IT-companies to set up a transnational collaboration scheme of higher level education in Databases. Together with colleagues from 5 European countries he was conducting projects supported by the European Union on state-of-the teaching and hands-on labs on database technology.

Prof. Laux received the 2012 Research Award from Reutlingen University and he is an IARIA fellow.
Resources available

https://www.smiffy.de/dbkda-2021/

1. Slideset
2. Exercises
3. Command refcard
4. Example datasets

---

1. All materials copyright 2017, 2018, 2019, 2020, 2021 by Andreas Schmidt
Outlook

- Introduction
- Functionality Overview
- Filter & Pipes Architecture
- Command Overview Part I
- Exercise I: Start solving a criminal case using the shell
- Command overview Part II
- Exercise II: Solve the criminal case from Exercise I
- sed & awk
- Summary & Outlook
Why Should I use these Tools (Coreutils)?

With R and Python, there exist great tools that can perform the same job (and much more)

- It's already on your computer and nothing needs to be installed\(^1\)
- You don’t need to learn a programming language
- You don’t need an editor, compiler or interpreter
- Low main memory footprint
- You got first results after 20 sec.
- Intuitive iterative development cycle (add filter by filter) ... like lego blocks
- It makes fun !!!!

\(^1\) if you have a Linux or Mac-computer. Windows users have to install cygwin to use these tools.
Filter and Pipes Architecture

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

• In-/Output Redirection
  • | : Pipe operator: Connect STDOUT of a command with STDIN of the next command
  • > : Redirect Standard Output (into file)
  • < : Redirect Standard Input (from file)
  • 2> : Redirect Standard Error (into file)
  • >> : Redirect Standard Output (append into file)

• Example:

  `cut -d, -f1 city.csv | sort | uniq -c | \`
  `sort -nr | awk '$1>1' > result.txt`
Retrieving the names of cities which have „name siblings“

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

Binjai
Hsinchu
Zhuhai
Jinxi
Reynosa
Livonia
"Hpa an"
Paterson
Kaifeng
Orlando
Brescia
Tepic
...
Another example: Word count

grep '[A-Za-z]+' -Eo moby-dick.txt | \
    tr 'A-Z' 'a-z' | \
    sort | uniq -c | sort -nr | less

The Project Gutenberg EBook of Moby Dick or The Whale...

the project gutenberg ebook of moby dick or the whale...

14715 the 6742 of 6517 and 4805 a 4707 to 4241 in 3100 that 2536 it 2532 his

4805 a 2 aback 2 abaft 3 abandon 7 abandoned 1 abandonedly 2 abandonment 2 abased

...
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 (some commands like join, comm expect a „-“ character as parameter, if 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 can’t be explained in detail. To get an overview of the possibilities of a command, simple type

```
man command
```

• Example:

```
man head
```
head - output the first part of files

head [OPTION]... [FILE]...

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)
cat command

• Print content of file to STDOUT
  ```
cat HelloWorld.java
  ```

• Concatenate files and writes them via redirection (>) to a file
  ```
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-search-words.txt
  Obama
  Climate
  CTRL-D
  ```

• More example:
  ```
  www.smiffy.de/dbkda-2021/command-examples/cat,%20head,%20tail,%20less,%20wc
  ```
head/tail/wc command

- **head**: view first \( n \) lines or skip last \( n \) lines of a file.
  - View first 5 lines from file:
    
    ```
    head -n5 city.csv
    ```
  - Print all but the last 20 lines:
    
    ```
    head -n -20 city.csv
    ```

- **tail**: view last \( n \) lines or start from line \( n \)
  - View last 10 lines of a file
    
    ```
    tail -n 10 city.csv
    ```

- **wc**: Count the number of lines, words\(^1\) and bytes
  
  ```
  wc city.csv
  ```

---

1. from the manual page: A word is a non-zero-length sequence of characters delimited by white space.
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
  - nd: scroll forward n lines (i.e. 20n)
  - mb: scroll backwards m lines
  - ng: Goto line <n>
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
grep command

- Print lines matching pattern (case sensitive)
  
  \texttt{grep USA city.csv}

- Print lines containing the regular expression (City starting with 'S', ending with 'g')
  
  \texttt{grep -E 'S[a-z]+g,' city.csv}

- Print only lines, not containing the String NULL
  
  \texttt{grep -v NULL city.csv}

- Print name of files which contain the pattern 'Agassi'
  
  \texttt{grep Agassi bbcsport/tennis/*.txt}

\textbf{when multiple files are queried, the filename is part of the result (<filename>:<line matching pattern>)}
Search

- Print only matching part (i.e. 'Salzburg' instead of whole line)
  
  ```
  grep -E -o 'S[a-z]+g' city.csv
  ```

- Look for lines containing words from a file

  ```
  grep -f grep-search-words.txt -E newsCorpora.csv
  ```

  - file: grep-search-words.txt
    - Obama
    - Climate

More example:

  https://www.smiffy.de/dbkda-2021/command-examples/grep
File operations

- Split file by number of rows (here, after each 10 lines, all generated files have the prefix tmp/city-part-)

```
csplit -k -f tmp/city-part- city.csv 10 {*}  
```

- Split file at every empty line (all generated files have the prefix tmp/person)

```
csplit -f tmp/person address-book.txt '/^ *$/' {*}  
```
File operations

- Print selected parts of lines from each file to standard output.
  
  `cut -d',' -f1,4 city.csv`

  Column separator  Output columns 1 and 4

- Output bytes 10 to 20 from each line
  
  `cut -b10-20 data.fixed`

- Output bytes 1-5 and starting from position 21 to the end of line:
  
  `cut -b1-5,20- data.fixed`

- More examples:
  
  `https://www.smiffy.de/dbkda-2021-ce/cut,%20paste`
Summary of Fundamental File Operations

- cat
- split/csplit
- cut
- paste
- tee

STDIN  STDOUT
And now you are prepared for a ...

**thrilling exercise** * !!!!!!

Go to the page www.smiffy.de/dbkda-2021, open Exercise I and start solving the criminal case ...

(*) command line murders by Noah Veltman, https://github.com/veltman/clmystery
mystery File/Directory Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Änderungsdatum</th>
<th>Typ</th>
<th>Größe</th>
</tr>
</thead>
<tbody>
<tr>
<td>interviews</td>
<td>16.04.2021 17:19</td>
<td>Dateiordner</td>
<td></td>
</tr>
<tr>
<td>memberships</td>
<td>16.04.2021 17:19</td>
<td>Dateiordner</td>
<td></td>
</tr>
<tr>
<td>streets</td>
<td>16.04.2021 17:19</td>
<td>Dateiordner</td>
<td></td>
</tr>
<tr>
<td>crimescene</td>
<td>16.04.2021 17:19</td>
<td>Datei</td>
<td>417 KB</td>
</tr>
<tr>
<td>people</td>
<td>16.04.2021 17:19</td>
<td>Datei</td>
<td>219 KB</td>
</tr>
<tr>
<td>vehicles</td>
<td>16.04.2021 17:19</td>
<td>Datei</td>
<td>486 KB</td>
</tr>
</tbody>
</table>

starting point
head mystery/people

To go to the street someone lives on, use the file for that street name in the 'streets' subdirectory. To knock on their door and investigate, read the line number they live on from the file. If a line looks like gibberish, you're at the wrong house.

<table>
<thead>
<tr>
<th>NAME</th>
<th>GENDER</th>
<th>AGE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alicia Fuentes</td>
<td>F</td>
<td>48</td>
<td>Walton Street, line 433</td>
</tr>
<tr>
<td>Jo-Ting Losev</td>
<td>F</td>
<td>46</td>
<td>Hemenway Street, line 390</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Annabel Fuglsang</td>
<td>M</td>
<td>40</td>
<td>Haley Street, line 176</td>
</tr>
<tr>
<td>Diego Michan</td>
<td>M</td>
<td>74</td>
<td>Wyola Place, line 25</td>
</tr>
</tbody>
</table>
File mystery/streets/Haley_Street, lines 174 - 179

... pinto simile fuzing pestering neutralized atriums

irradiates liquidates flimflams dispossessed

SEE INTERVIEW #871877

balmy metamorphosis nervier pilfered

proofreaders steeping editorialized solutions

- Interviews:

  $ ls mystery/interviews/interview-* | head -n5
  mystery/interviews/interview-000296
  mystery/interviews/interview-00448418
  ...
  mystery/interviews/interview-871877

  $ cat mystery/interviews/interview-871877
  Mr. Fuglsang is male and has brown hair ...
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 alphabetic, 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 the entries in file
    ```bash
    sort member-list.txt
    ```
  - sort the entries in file (Format: <first-name> <last-name>) by second column
    ```bash
    sort -t ' ' -k 2 member-list.txt
    ```
sort - examples

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

\[
\text{sort} \ -t, \ -k2,2 \ -k4,4nr \ \text{city.csv}
\]

field separator: ,
numeric (-n), descending (-r)
second sort criteria from column 4 to column 4
first sort criteria from column 2 to column 2

- More example:

  https://www.smiffy.de/dbkda-2021/command-examples/sort
Compare Operator

- `comm` - compare two sorted files line by line

- Options:
  - `-1`: suppress column 1
  - `-2`: suppress column 2
  - `-3`: suppress 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 comparison can be specified (first n chars, skip first m chars)
- options:
  - -c: count number of occurrences
  - -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

More example:

https://www.smiffy.de/dbkda-2021/command-examples/comm,%20uniq
Exercise Part II and III

continue solving the case from the first exercise (Exercise II) ...

... still thrilling ;-)...

... and with your actual knowledge, you can also try Exercise III ...
sed Principles

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

**s**: substitute

- Replace all occurrences of D with GER
  
  ```bash
  sed 's/\bD\b/GER/g' city.csv > city2.csv
  ```

- Replace all occurrences of NULL in a line with `\N` (Inplace Substitution)
  
  ```bash
  sed -i 's/\bNULL\b/\N/g' city.csv
  ```

- Replace „Stuttgart“ with „Stuttgart am Neckar“ (extended regexp)
  
  ```bash
  sed -E '/^Stuttgart/ s/^[^,]+/\1 am Neckar/' 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)
  
  ```bash
  sed -n 10,20p city.csv
  sed -n '5,10p;23p;56,71p' city.csv
  ```
sed Examples

i: insert

• Insert dataset about Karlsruhe at line 2
  ```
  sed '2i Karlsruhe,D,"Baden Wuerttemberg",312005,49.0,6.8' city.csv
  ```

d: delete

• delete the city 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
  ```
sed Examples

c: change

• Replace entry of Biel
  
  `sed '/^Biel,/,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`

r: read file

• insert the content from file city-D.csv starting at line 3
  
  `sed '3 r city-D.csv' city.csv'`
awk

- like sed, but with powerful programming language
- filter and report writer
- flexible record definition (i.e. line with columns, record with fields, ...)
- full programming language, support for associative arrays
- structure: one or multiple `pattern { action }` blocks
- special BEGIN, END pattern match before the first record is read and after the last record is read
- Access to column values via $1, $2, ... variables ($0: whole record)
- Examples:

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

- Calculating average population

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

```bash
# script: average.awk
BEGIN { sum = 0
    num = 0
}
$4 != "NULL" {
    sum += $4
    num++
}
END { print "Average population: "sum/num }
```
awk

• 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
  • OFS: output field separator
  • FPAT: Field pattern (alternative way to specify a field instead of use of FS)
  • FILENAME: contains the file that is actually read
awk Example: Multi-Line Records

Input

Andreas Schmidt
KIT
Germany

Lisa Ehrlinger
JKU
Austria

Fritz Laux
University Reutlingen
Germany

cat adress.txt | awk -f rec2csv.awk

begin {
  FS=\n
  RS=\n
  OFS=",

  ORS=\n
}

{ $1=$1
  print $0
}

set input and output separators

give awk a hint that anything has changed

Output

Andreas Schmidt, KIT, Germany
Lisa Ehrlinger, JKU, Austria
Fritz Laux, University Reutlingen, Germany
And again its time for crime* ... 

As homework: You have solved the case, but there is room for improvement for future cases. So continue with Exercise IV ...

(*) command line murders by Noah Veltman, https://github.com/veltman/clmystery
Commands not Covered (not complete)

- **xargs**: build and execute command lines from standard input
  
  ```bash
  grep -l Agassi bbcSPORT/tennis/*.txt | xargs grep -l Federer
  ```

- **tr**: translate, squeeze, and/or delete characters from standard input, writing to standard output.
  
  ```bash
  tr 'A-Z' 'a-z' < moby-dick.txt
  ```

- **paste**: merge lines of files
  
  ```bash
  paste -d',' col1.txt col2.txt col3.txt > col_1-3.txt
  ```

- **find**: search for files in a directory hierarchy
  
  ```bash
  find ./misc -name /*.txt -print
  ```

- **join**: join lines of two files on a common field
join Example

city.csv

Aachen, D, "Nordrhein Westfalen", 247113, NULL, NULL
Aalborg, DK, Denmark, 113865, 10, 57
Aarau, CH, AG, 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, 52496:
Algeria, DZ, Algiers, Algeria, 2381740, 291830
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
Summary & Outlook

• Summary
  • Powerful filter and pipes architecture
  • Allows easy incremental development
  • Suitable for structured and unstructured data, ETL process
• Outlook
  • Utility make to handle dependencies between files
  • bash control flow elements like conditional execution, loops
  • bash functions
  • Seamless visualization using gnuplot
  • Easily extensible with own filters in any language
References & Further Readings

- gnuplot homepage: http://www.gnuplot.info/