Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

The Eighth International Conference on Data Analytics

September 22, 2019 to September 26, 2019 - Porto, Portugal

Title: Analyzing the Enron Corpus with the Shell

Andreas Schmidt

(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/data-analytics-2019/ [1]

- Slideset
- Exercises
- Command refcard
- Example datasets

1. all materials copyright 2017, 2018, 2019 by andreas schmidt

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Outlook

- Overview
- Search and Inspect
- File operations
- Transformations
- Stream Editor sed
- Anlyzing the Enron-files
- Summary

+ 2 hands on exercices

Objective of this Tutorial

Present you

- ... Concept of Filters and Pipes (Interactive, iterative workstyle)
- ... most useful Shell tools/programs in the context of the Enron dataset
- ... Practical exercises - get your fingers dirty ;-)

The UNIX philosoph

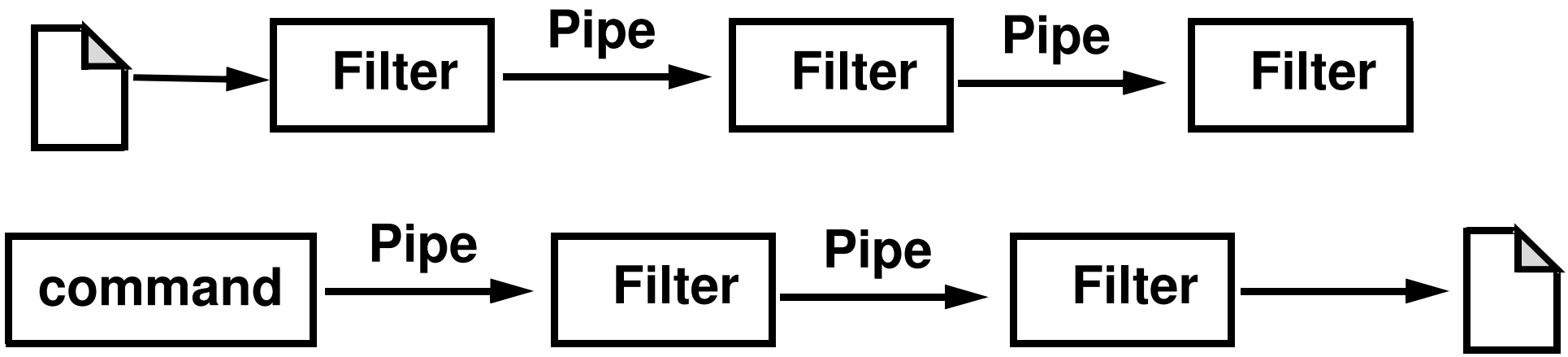
Peter H. Salus in A Quarter-Century of Unix (1994):[1]

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

[1] Raymond, Eric S. (2003-09-23). "Basics of the Unix Philosophy". The Art of Unix Programming. Addison-Wesley Professional. ISBN 0-13-142901-9.

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



Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

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
```

Overview over Operations

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

How an email looks like

```
Message-ID: <8150320.1075849854060.JavaMail.evans@thyme>
Date: Tue, 5 Dec 2000 09:42:00 -0800 (PST)
From: shona.wilson@enron.com
To: leslie.reeves@enron.com, mike.jordan@enron.com, james.new@enron.com,
  peggy.hedstrom@enron.com
Subject: global standards needs assessment
Cc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
X-From: Shona Wilson
X-To: Leslie Reeves, Mike Jordan, James New, Peggy Hedstrom
X-cc: Kelly Tisman, Mike Perun, Jefferson D Sorenson, Sally Beck, Brent A Price, Christy Lobusch
X-bcc:
...

Here is the draft of the needs assessment for global standards.  Please give
me your input.  Kelly is devoting 100% of her time to putting together an
access database to report these standards.
To: ich@smiffy.de
Also, here is a consolidated view of A01 for 12 /04.
```

How an email looks like

Header

```
Message-ID: <8150320.1075849854060.JavaMail.evans@thyme>
Date: Tue, 5 Dec 2000 09:42:00 -0800 (PST)
From: shona.wilson@enron.com
To: leslie.reeves@enron.com, mike.jordan@enron.com, james.new@enron.com,
  peggy.hedstrom@enron.com
Subject: global standards needs assessment
Cc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
X-From: Shona Wilson
X-To: Leslie Reeves, Mike Jordan, James New, Peggy Hedstrom
X-cc: Kelly Tisman, Mike Perun, Jefferson D Sorenson, Sally Beck, Brent A Price, Christy Lobusch
...

Here is the draft of the needs assessment for global standards.  Please give
me your input.  Kelly is devoting 100% of her time to putting together an
access database to report these standards.
To: ich@smiffy.de
Also, here is a consolidated view of A01 for 12 /04.
```

How an email looks like

Header

```
Message-ID: <8150320.1075849854060.JavaMail.evans@thyme>
Date: Tue, 5 Dec 2000 09:42:00 -0800 (PST)
From: shona.wilson@enron.com
To: leslie.reeves@enron.com, mike.jordan@enron.com, james.new@enron.com,
  peggy.hedstrom@enron.com
Subject: global standards needs assessment
Cc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
X-From: Shona Wilson
X-To: Leslie Reeves, Mike Jordan, James New, Peggy Hedstrom
X-cc: Kelly Tisman, Mike Perun, Jefferson D Sorenson, Sally Beck, Brent A Price, Christy Lobusch
...
```

Body

```
Here is the draft of the needs assessment for global standards.  Please give
me your input.  Kelly is devoting 100% of her time to putting together an
access database to report these standards.
To: ich@smiffy.de
Also, here is a consolidated view of A01 for 12 /04.
```

How an email looks like

Header

```
Message-ID: <8150320.1075849854060.JavaMail.evans@thyme>
Date: Tue, 5 Dec 2000 09:42:00 -0800 (PST)
From: shona.wilson@enron.com
To: leslie.reeves@enron.com, mike.jordan@enron.com, james.new@enron.com,
  peggy.hedstrom@enron.com
Subject: global standards needs assessment
Cc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: kelly.tisman@enron.com, mike.perun@enron.com, jefferson.sorenson@enron.com,
  sally.beck@enron.com, brent.price@enron.com,
  christy.lobusch@enron.com
X-From: Shona Wilson
X-To: Leslie Reeves, Mike Jordan, James New, Peggy Hedstrom
X-cc: Kelly Tisman, Mike Perun, Jefferson D Sorenson, Sally Beck, Brent A Price, Christy Lobusch
...
```

Header-Field

Body

```
Here is the draft of the needs assessment for global standards.  Please give
me your input.  Kelly is devoting 100% of her time to putting together an
access database to report these standards.
To: ich@smiffy.de
Also, here is a consolidated view of A01 for 12 /04.
```

Enron dataset

- Source: https://www.cs.cmu.edu/~enron/
- ~ 500.000 emails from 150 employees (mostly managers)
- Format (output generated `find maildir/ -name *. -print | sort -R`):

```
maildir/symes-k/sent/708.
maildir/farmer-d/logistics/907.
maildir/sager-e/notes_inbox/311.
maildir/rodrique-r/sent/174.
maildir/delainey-d/_sent_mail/180.
maildir/steffes-j/inbox/25.
maildir/dasovich-j/all_documents/11188.
maildir/gang-l/deleted_items/80.
maildir/whalley-l/discussion_threads/1209.
maildir/symes-k/all_documents/2487.
maildir/mclaughlin-e/eol___tagg/98.
maildir/taylor-m/inbox/58.
```

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

General comments

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 -n10 maildir/arnold-j/inbox/19.
cat -n maildir/arnold-j/inbox/19. | head -n5
```

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

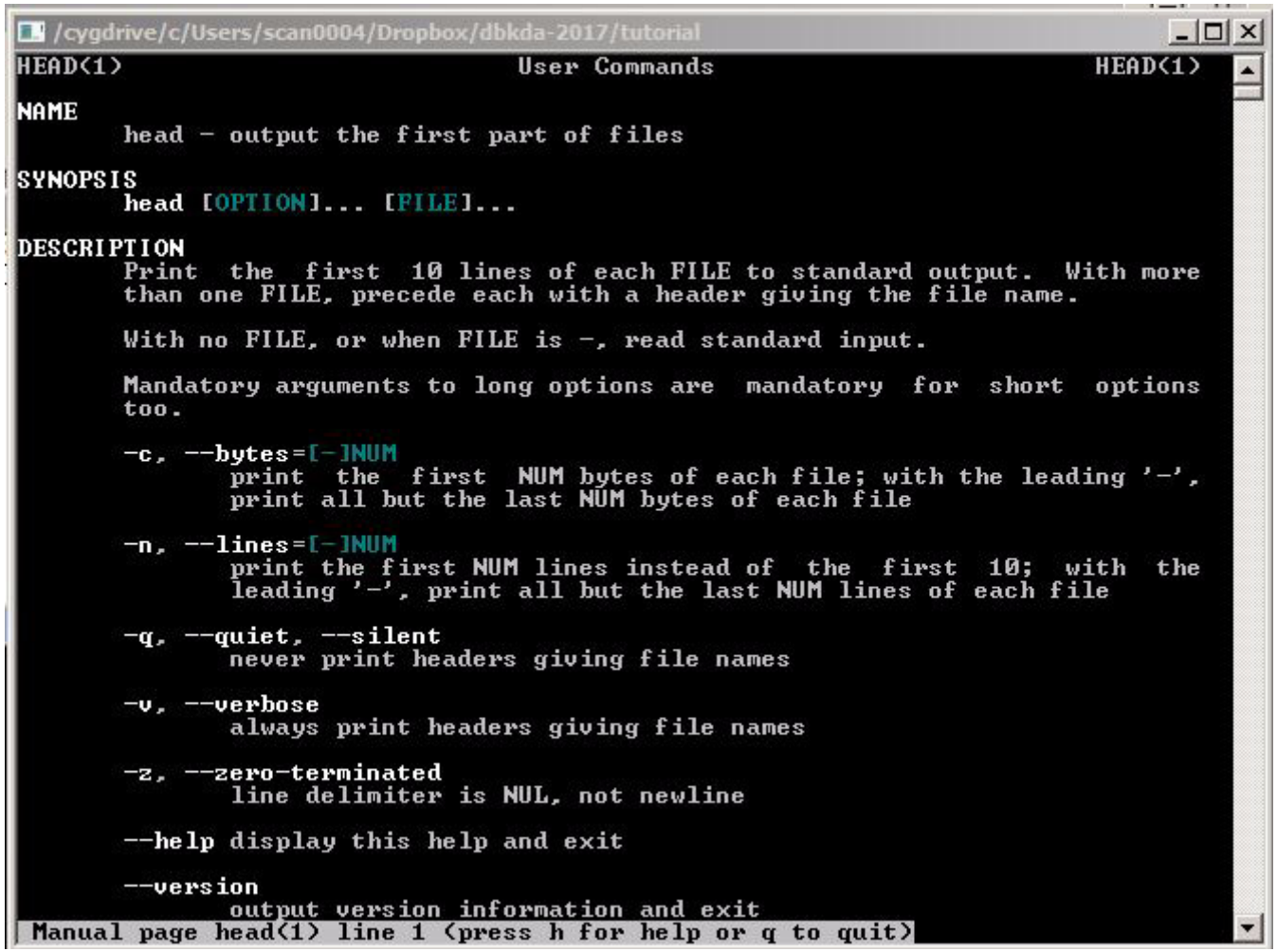
```
man command
```

Example:

```
man head
```

Command refcard:

```
www.smiffy.de/data-analytics-2019/tutorial-overview-commands.pdf
```


/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)

Karlsruhe Institute of Technology

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

File Inspection

- Show content of a file / Concatenate files

```
cat maildir/arnold-j/inbox/19.
```

- Concatenate files and print them to STDOUT

```
cat maildir/arnold-j/inbox/* > arnolds-inbox.mail
```

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

```
cat -n maildir/arnold-j/inbox/19.
```

- Create a file with input from STDIN:

```
cat > search-fields.txt
Moody
Standard & Poor
Justice Department
CTRL-D
```

File Inspection

- View first 5 lines from file:

```
head -n5 ./inbox/19.
```

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

```
cat -n ./inbox/19.| tail -n4
```

- View content of file, starting from line 40:

```
tail -n +40 ./inbox/19.
```

- Print all but the last 2 lines:

```
head -n -2 /inbox/19.
```

- Count the number of lines, words and bytes

```
wc ./inbox/19.
```

- Count the number of lines

```
wc -l ./inbox/19.
```

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 ./inbox/19.
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, case insensitive with option -i))

```
grep John ./inbox/149.
```

- Search multiple files (if multiple files are searched, output contains filename and matching lines):

```
grep John ./inbox/*.
```

- Print lines containing the regular expression (emails from .edu-domains)

```
grep -E '[A-Za-z.]+@[A-Za-z.]+\.edu' ./inbox/1021.
```

- Print matching lines in a binary file

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

- Prefix each line of output with the line number

```
grep -n '^From:' inbox/279.
```

- Suppress the output of the filename (remember, if multiple files are asked, the filename is preceded by default)

```
grep -n -h '^From:' inbox/*.
```

- Output the name of files matching a pattern (instead of content)

```
grep -E -l Moody maildir/arnold-j/inbox/*.
```

- Output the name of all files which have no Cc header-field

```
grep -L '^Cc:' maildir/*/*/*.
```

Search

- Look for lines containing words from file

```
grep -f search-fields.txt inbox/*
```

  - file: **search-fields.txt**

```
Moody
Standard & Poor
Justice Department
```

- Count the number of lines, in which one of the words from search-fields appear (outputformat: <filename>:<no of matching lines>)

```
grep -c -f search-fields.txt inbox/*
```

- Stop search, after n-th occurence of pattern is found (here: n is 1)

```
grep -m1 '^To:' maildir/shackleton-s/inbox/279.
```

- Extract all uppercase words in a text (each word in a separate line):

```
grep -E -o '[A-Z][a-z]+' ../maildir/arnold-j/*/*
```

- Remove line comments from source file
  (`// This is a comment - please remove me`)

```
grep -v '^ *//' src/levensthein.cpp
```

  for multi-line comments, see command `sed`

- Print not only the line, but also the n follwing lines (here: n is 4)

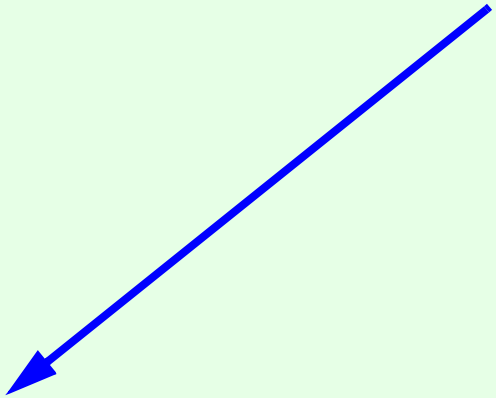
```
grep -A4 To: ../maildir/arnold-j/*/*
```

grep & xargs

- xargs - build and execute command lines from standard input
- Example (return all files, which contain the words 'brilliant' and 'corrupt'):

```
grep -l brilliant ./docs/*.txt | xargs grep -l corrupt
```

return filename instead of line matching pattern



expands to:

```
grep -l corrupt docs/file1.txt docs/file7.txt docs/file9.txt ...
```

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)
```

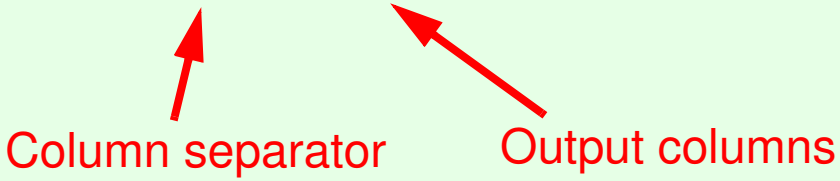
  - Runtime:

```
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.
```

Split/Merge Files Vertically

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

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



- Output bytes 10 to 20 from each line

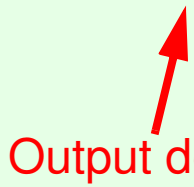
```
cut -b10-20 data.fixed
```

- Output characters 1 to 3, and 5 and 8 from each line

```
cut -c1-3,5,8 utf8-table.txt
```

- Merge lines of files (columnwise)

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



File operations - Split files horizontally

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

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

- Split file at every Form-Feed character (each page in a separate file)

```
split -t$'\14' --lines=1 IIBB_Notenspiegel_20172.txt
```

alternate record separator (instead of newline)

csplit

- Split a file into sections determined by context lines
- Usage:

```
csplit [options] file pattern(s)
```

- Pattern Semantic:
  - /pattern/: copy up, to but not including the matching line
  - %pattern%: skip to, but not including the matching line
- Example:

```
csplit notes_inbox/2066. '/^$/' # split header from body

csplit notes_inbox/2066. '%^$%' '/^--* Forwarded by/' {*} \
       --prefix notes_inbox/2066.
```

repetition factor

matches empty line

File operations - Split files horizontally

```
csplit ./all_documents/1374. %^To:% /^Subject:/ --prefix 1374.
```

- file `all_documents/1374.`

```
Message-ID: <13404045.1075855777773.JavaMail.evans@thyme>
Date: Wed, 27 Sep 2000 05:47:00 -0700 (PDT)
From: scott.earnest@enron.com
To: anjali.abraham@enron.com, robert.bonin@enron.com,
        kara.boudreau@enron.com,ary.denson@enron.com,
       matt.brown@enron.com, michelle.bruce@enron.com
Subject: Global Products - Houston Organizational Announcement
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
...
```

```
$ cat 1374.00
To: anjali.abraham@enron.com, robert.bonin@enron.com,
        kara.boudreau@enron.com,ary.denson@enron.com,
       matt.brown@enron.com, michelle.bruce@enron.com
```

File operations - Split files horizontally

```
csplit ./all_documents/1374. %^To:% /^[A-Z][a-z]*:/ --prefix 1374.
```

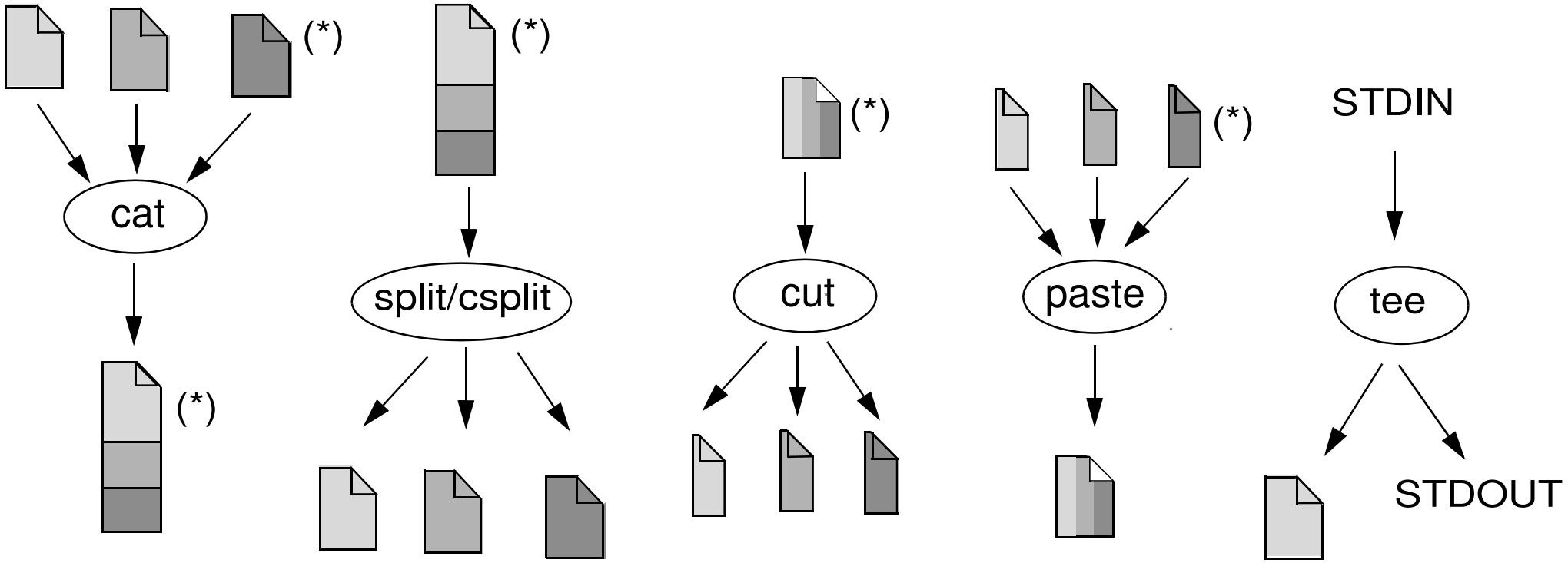
- file `all_documents/1374.`

```
Message-ID: <13404045.1075855777773.JavaMail.evans@thyme>
Date: Wed, 27 Sep 2000 05:47:00 -0700 (PDT)
From: scott.earnest@enron.com
To: anjali.abraham@enron.com, robert.bonin@enron.com,
        kara.boudreau@enron.com,ary.denson@enron.com,
       matt.brown@enron.com, michelle.bruce@enron.com
Subject: Global Products - Houston Organizational Announcement
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
...
```

more general pattern

```
$ cat 1374.00
To: anjali.abraham@enron.com, robert.bonin@enron.com,
        kara.boudreau@enron.com,ary.denson@enron.com,
       matt.brown@enron.com, michelle.bruce@enron.com
```

Summary File operations



*) or at most one from STDIN

Transformation - 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:
  - $\{c_1,c_2,c_3,c_4,c_5,c_6\}$
- Squeeze:
  - {aa...a -> a, xx...x -> x, \n\n...\n->\n}
- Predefined character class/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>

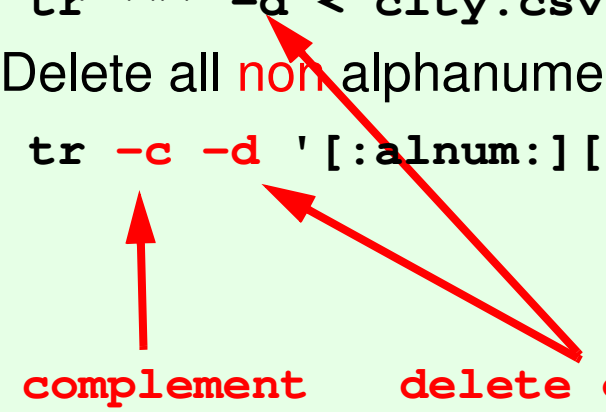
```
tr '\n' ' ' < short-story.txt > one-liner.txt
```

  - 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: ,)

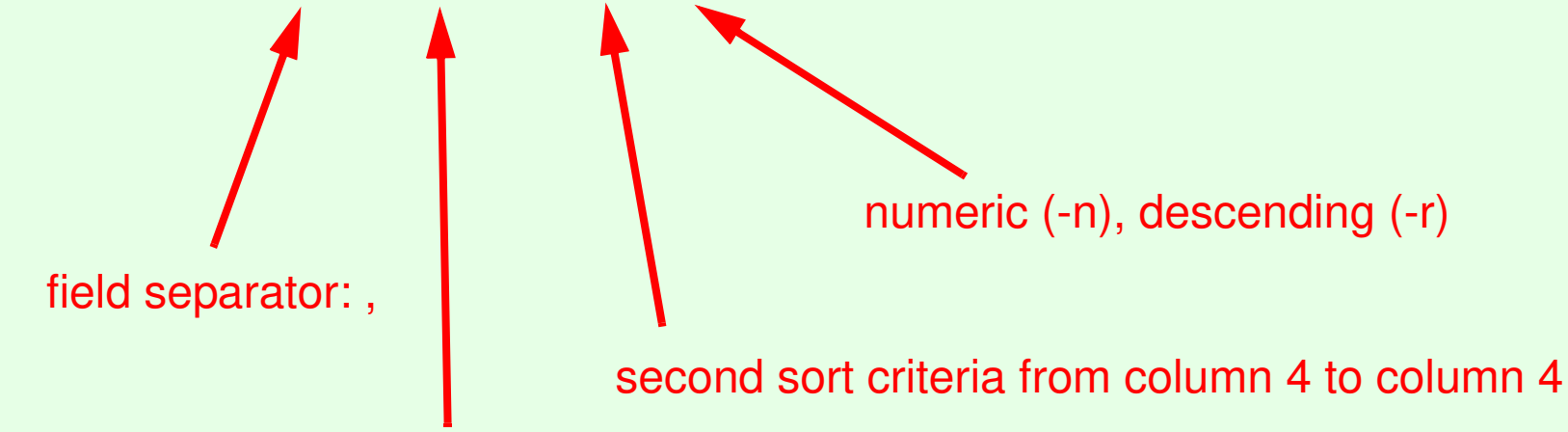
    ```
    sort -k2 -t',' city.csv
    ```
  - merge content of file1.txt and file2.txt and sort the result

    ```
    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, -k1.2,1.3 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
```

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 !!!

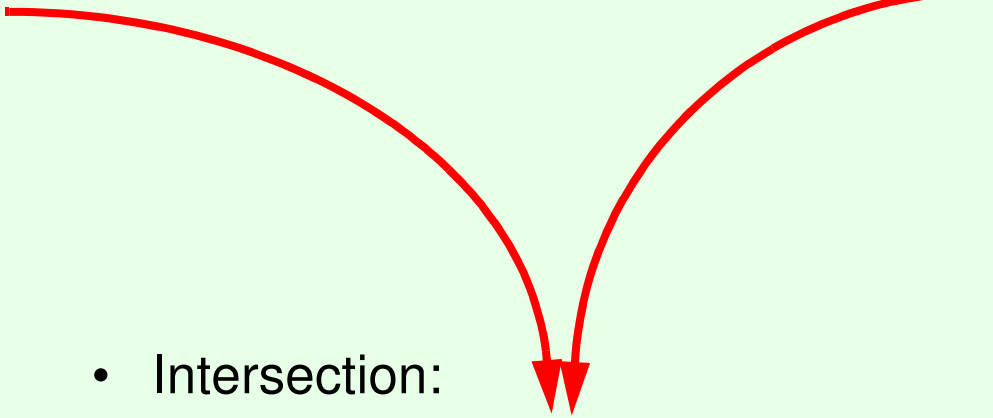
uniq - example

- 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 I - (Duration 10 - 15 min.)

see handout ...

(http://www.smiffy.de/data-analytics-2019/bash-exercise-1.pdf)

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 (-o option)
- Example:

country is already sorted by 2. column

```
sort -k2.2 -t, city.csv | join -t, -12 -22 - country.csv \
                                -o1.1,2.1,1.3,1.4
```

sort by the column to join

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

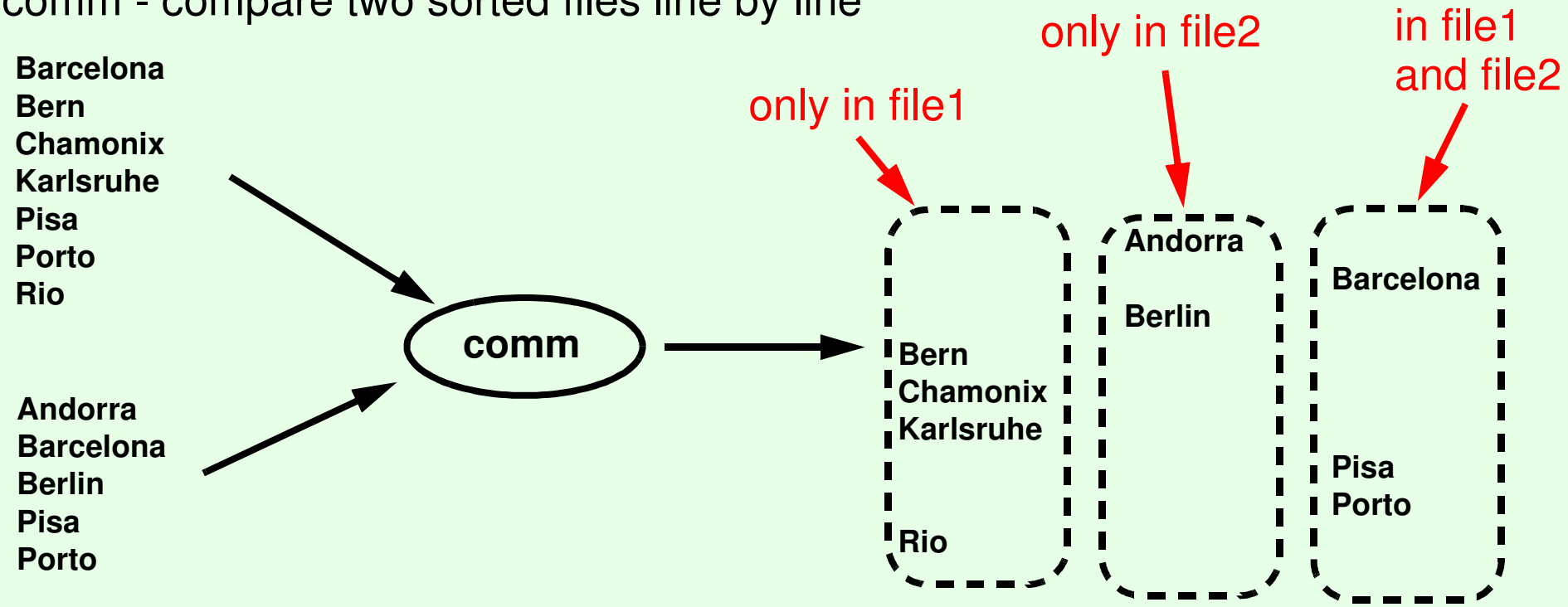
```
...
Germany,D,Berlin,Berlin,356910,83536115
Djibouti,DJI,Djibouti,Djibouti,22000,42
Denmark,DK,Copenhagen,Denmark,43070,524
Algeria,DZ,Algiers,Algeria,2381740,2918
Spain,E,Madrid,Madrid,504750,39181114
...
```

```
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

not semantic with sorted input !!!

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 a single 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 '/\bStuttgart\b/ s/^(Stuttgart)/\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
```

sed

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

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

- **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
```

- ...

Exercise II - (Duration 15 - 20 min.)

see handout ...

(http://www.smiffy.de/data-analytics-2019/bash-exercise-2.pdf)

Processing the Enron dataset

Some statistics:

- How many emails do we have?

```
find maildir -name *. -print| wc -l
```

- How many emails are available from the different persons?

```
find maildir -name '*.' -print| sed -r 's#maildir/([^/]+)/.*#\1#'|\
sort | uniq -c| sort -n
```

- Write convenience function:

```
allmails() {
    find maildir -name '*.' -print
}
```

- now we can type (... and ask how many emails are in jones' mailbox):

```
allmails | grep 'maildir/jones' | wc -l
```

- Extract all email adresses from the email files

```
find maildir -name '*.' -print | \
  xargs grep -h -E -o '[-A-Za-z._0-9]+@[-A-Za-z._0-9]+\.[a-z]+\b' |
  tr 'A-Z' 'a-z' | sort | uniq > all-emails-from-files.txt
```

- look for the email addresses of the 160 enron employees from the dataset

```
allmails | cut -d/ -f2| cut -d- -f1| awk '{print $1"@enron.com"}' \
   > enrons160-email-pattern.txt
allmails | xargs grep -i -o -f enrons160-email-pattern.txt -h | \
   tr 'A-Z' 'a-z' | sort | uniq
```

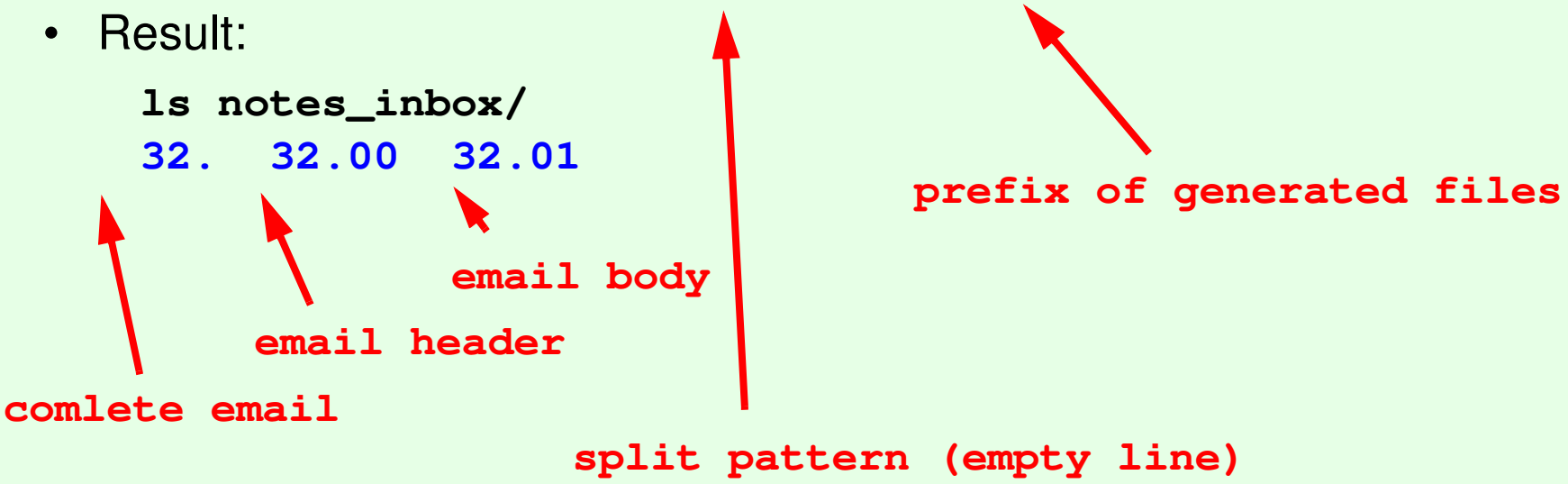
- and many many more ...

- Split an email into header and body :

```
csplit notes_inbox/32. /^$/ -f notes_inbox/32.
```

- Result:

```
ls notes_inbox/
32.   32.00   32.01
```



- Do this for all mails:

```
allmails | xargs -I @ csplit @ /^$/ -f @
```

- Result:

```
ls maildir/zufferli-j/deleted_items/119*
maildir/zufferli-j/deleted_items/119.
maildir/zufferli-j/deleted_items/119.00
maildir/zufferli-j/deleted_items/119.01
```

- Rename files:

```
allmails  | xargs -I @ mv @01 @body
allmails  | xargs -I @ mv @00 @header
```

- Result:

```
ls maildir/zufferli-j/deleted_items/119*
maildir/zufferli-j/deleted_items/119.
maildir/zufferli-j/deleted_items/119.header
maildir/zufferli-j/deleted_items/119.body
```

Extract all recipients from an email:

```
sed -n '/^To:/,/^[A-Z]/ p' 32.header| \
   head -n -1 | \
   tr -d '\n\t' | \
   sed 's#To: ##'
ywang@enron.com, patti.sullivan@enron.com, phil-
lip.k.allen@enron.com, jane.m.tholt@enron.com,
mike.grigsby@enron.com
```

Generalize problem with a function (for all fields)

```
get_field() { # $1: pattern, #2: file
  sed -n "/^$1:/,/^[A-Z]/ p" $2 | head -n -1 | tr -d '\n\t' | \
    sed "s#^$1: ##"
}
```

Email Header Format

```
$ cat maildir/allen-p/notes_inbox/32.header
Message-ID: <3351977.1075855679526.JavaMail.evans@thyme>
Date: Tue, 12 Dec 2000 23:04:00 -0800 (PST)
From: critical.notice@enron.com
To: ywang@enron.com, patti.sullivan@enron.com, phillip.k.allen@enron.com,
        jane.m.tholt@enron.com, mike.grigsby@enron.com
Subject: New Notice from Transwestern Pipeline Co.
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: critical.notice@Enron.com
X-To: ywang@Enron.com, Patti.Sullivan@Enron.com, Phillip.K.Allen@Enron.com,
jane.m.tholt@Enron.com, Mike.Grigsby@Enron.com
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Dec2000\Notes Folders\Notes inbox
X-Origin: Allen-P
X-FileName: pallen.nsf
```

Using function:

```
$ get_field From maildir/allen-p/notes_inbox/32.header
critical.notice@enron.com

$ get_field Date maildir/allen-p/notes_inbox/32.header
Tue, 12 Dec 2000 23:04:00 -0800 (PST)
```

Putting all together:

- Target: tsv-Files with the following format:
  - send_report.tsv

```
FROM                          TO                          TYPE   MESSAGE-ID
critical.notice@enron.com ywang@enron.com          To     <3351977.1075855679526@thyme>
critical.notice@enron.com patti.sullivan@enron.com To <3351977.1075855679526@thyme>
critical.notice@enron.com phillip.k.allen@enron.com To <3351977.1075855679526@thyme>
critical.notice@enron.com jane.m.tholt@enron.com To     <3351977.1075855679526@thyme>
critical.notice@enron.com mike.grigsby@enron.com To     <3351977.1075855679526@thyme>
...
```

  - email.tsv

```
SUBJECT                 MESSAGE-ID                          DATE                  LOCATION
"New Notice from ..."  <3351977.1075855679526@thyme> "2000-12-13 08:04" 5427.body
...
```

Putting all together (2)

- File process-single-email.sh

```
#!/bin/sh

. ./my-functions.sh                # definition of get_field()

echo processing $1 ...
export header=$1"header"
export from=`get_field From $header`
export id=`get_field Message-ID $header`
export recipients=`get_field To $header| tr ',' ' '`
for r in $recipients
do
  echo -e "$from\t$r\tTO\t$id" >> send_report.tsv
done
export subject=`get_field Subject $header`
export date=`get_field Date $header`
export date=`date --date="$date" +"%Y-%m-%d %H:%M"`
echo -e "$subject\t$from\t$id\t$date" >> email.tsv
exit 0
```

Putting all together (3)

- Process a single email:

```
sh ./process_single_email.sh maildir/bruno-b/inbox/2715.
```

- Iterate over all emails:

```
allmails | xargs.exe -n1 sh ./process_single_email.sh
```

- Now we can ask a number of intersting questions quite easily:
  - how many emails are there?

```
wc -l data/email.tsv
```

  - who has send the most emails to another person?

```
cut -f1,2 send_report.tsv | sort| uniq -c|sort -nr|head -n1
```

  - how many emails were sent each month?

```
cut -f3 email.tsv | grep '....-..' -o| sort |uniq -c|sort -k2
```

  - who are the persons who received the most emails?

```
cut -f2  data/send_report.tsv | sort| uniq -c| sort -n| tail
```

Visualization with Gnuplot

- how many emails were sent each month?

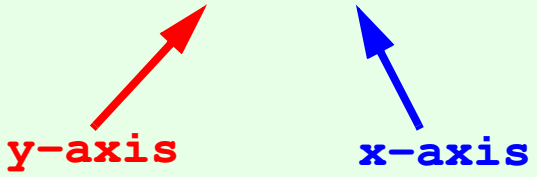
```
cut -f3 email.tsv | grep '....-..' -o| sort |uniq -c|sort -k2 \
  > emails_per_month.txt
```

- email-per-month.txt

```
...
6 2000-02
9 2000-03
11 2000-04
15 2000-05
15 2000-06
16 2000-07
26 2000-08
28 2000-09
22 2000-10
25 2000-11
...
```

- emails-per-month.gplt

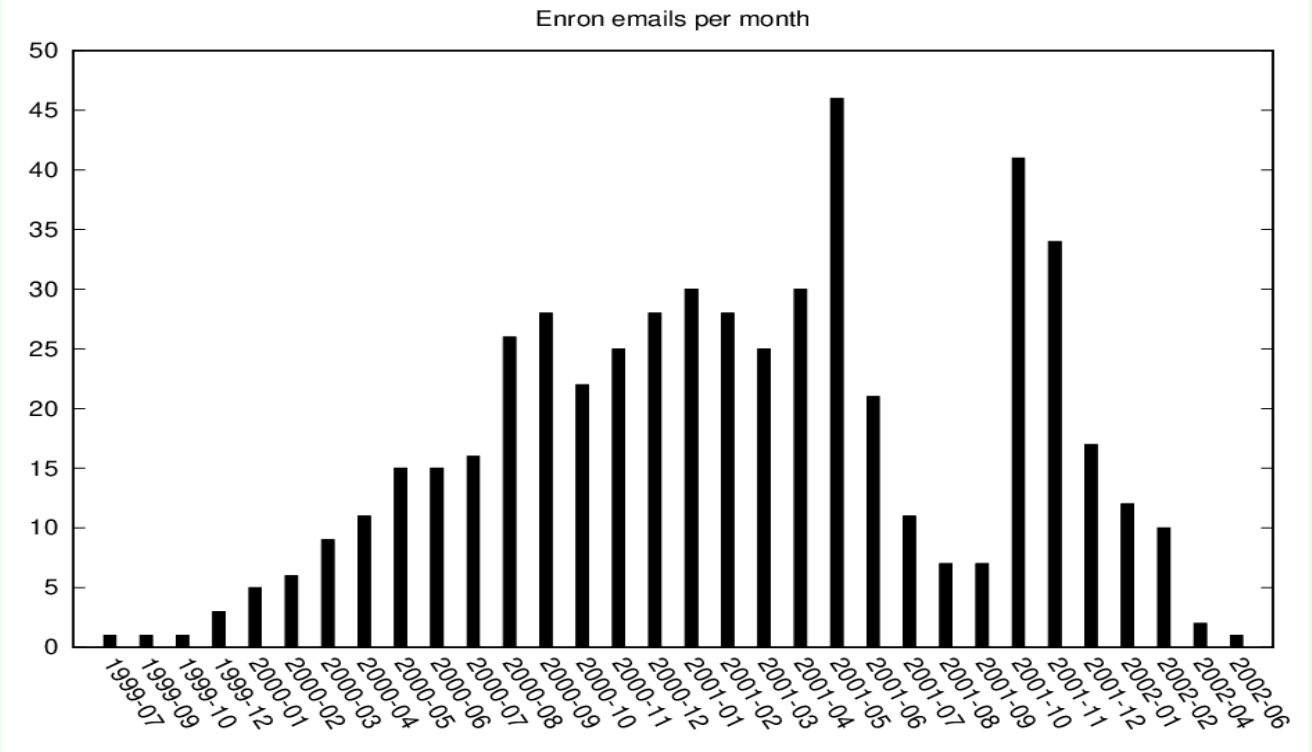
```
set terminal postscript
set output "plots/emails-per-month.ps"
set title "Enron emails per month"
set style fill solid
set style data histogram
set xtic nomirror rotate by -60
plot "emails-per-month.txt" \
      using 1:xtic(2) title ''
```



Visualization with Gnuplot

- Generate plot

  $ gnuplot emails-per-month.gplt



Visualization with Gnuplot

- file temp.dat

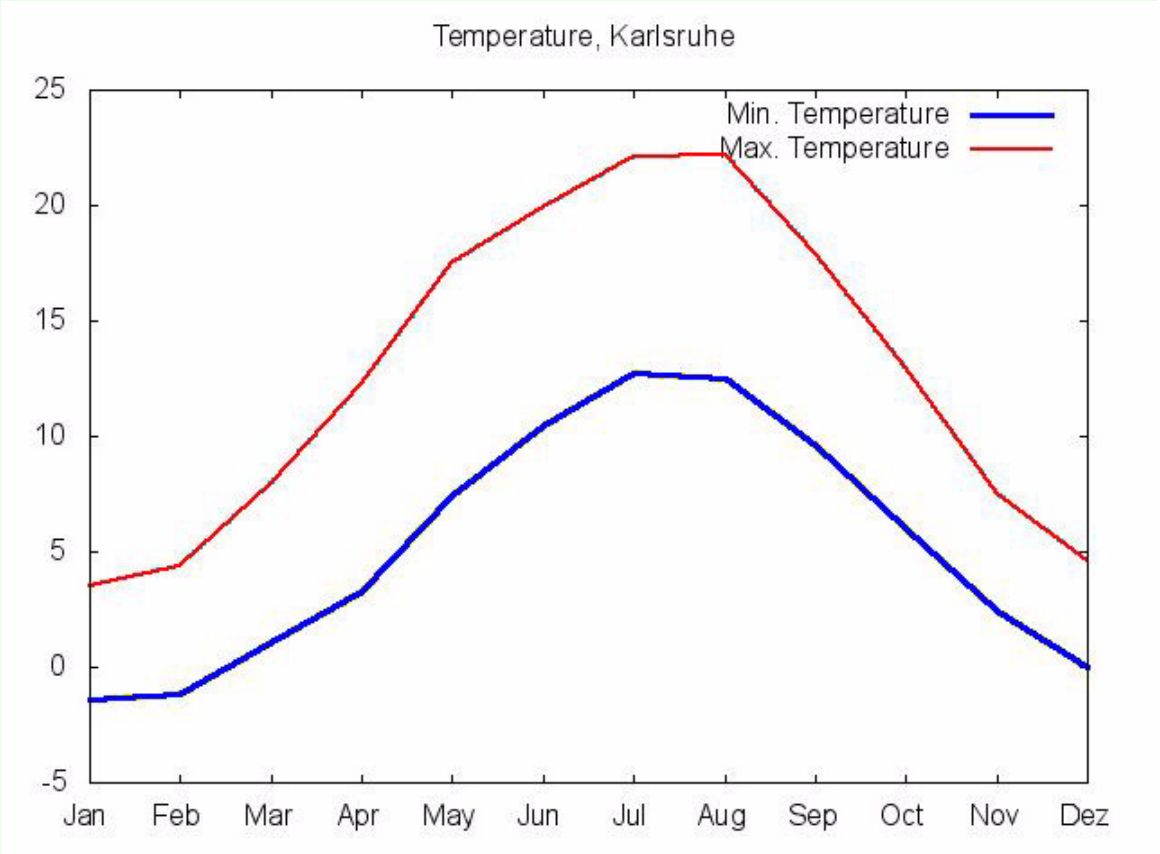
```
Month "Min. Temperature" "Max. Temperature"
Jan   -1.4                3.5
Feb   -1.2                4.4
Mar   1.1                 8.0
Apr   3.3                 12.3
May   7.4                 17.5
Jun   10.5                19.9
Jul   12.7                22.1
Aug   12.5                22.2
Sep   9.6                 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
```



Summary

- The shell offers a bunch of very usefull tools for data-processing
- Concept of filter and pipes allow iteratively development of complex transformations
- Free available, ready to use ... (no expensive import needed)
- Also powerful operations like joins, aggregations can be performed
- Extensible with programs in every programming language that could read from STDIN and write to STDOUT
- With the addition of gnuplot, sophisticated graphical plots can be drawn
- make (not handled in this tutorial), allows the formulation of dependency rules for the construction of complex workflows
- Potential Extension for the Enron Dataset: drawing graphs with GraphViz[1]

1. https://www.graphviz.org/