The The Tenth International Conference on Advances in Databases, Knowledge, (DBKDA 2018), May 20 - 24, 2018 – Nice/France

Tutorial: How to build a Search-Engine with Common Unix-Tools
Andreas Schmidt

Command Overview

Command	Description	Popular options
cat	concatenate files and print on the standard output	**-n**: number all output lines
comm	compare two sorted files line by line	**-1:** suppress column 1 (lines unique to FILE1) **-2:** suppress column 2 (lines unique to FILE2) **-3:** suppress column 3 (lines unique to FILE3) **--total** : output a summary
cut	remove sections from each line of files	**-d**<*delim*>: Use <*delim*> instead oft ab as field separator **-f**<*field-list*>*:* select only fields in <*field-list*> **--output-delimiter**=<*delim*> : use <*delim*> as output delimiter
grep	print lines matching a pattern	**-E**: support extended regexp **-P**: support Perl regexp (experimental) **-i**: ignore case **-v**: invert match **-h**: supress output of filenames **-o**: print only the matched part (output: one line per match) **-f**: obtain patterns from file **-l**: files with match **-L**: files without match **-c**: suppress normal output, instead count matching lines. **-m**<*num*>: Stop reading a file after <*num*> matching lines **--label**: Display input coming from STDIN, as input coming from file <file-name>. **--color=auto**: Display match in red color **-A<num>, -B<num>, -C<num>, -<num>**: Display <num> context lines **-a** : handle binary files as text **-n**: Add line n umber to match **-b**: add byte offset to text
head	output the first part of files	**-n**<*num*>*: print the first* <*num*> *lines* **-n** -<*num*>: print all but the last <*num*> lines
join	join lines of two files on a common field Remark: files must be sorted on join column	**-t**<*char*> : Use <*char*> as input, output separator **-1**<*field*> : join on this FIELD of file 1 **-2**<*field*> : join on this FIELD of file 2 **-o**<*format*> : obey <*format*> while constructing output line <*format*> *: filenum.field[[filenam.field][…]]*
man	an interface to the on-line reference manuals	
paste	merge lines of files	**-d**<char>: use <*char*> as output delimiter
seq	print a sequence of numbers	**-s**: separator (default: \n)
sort	sort lines of text files	**-n**: numeric sort **-r**: reverse sort

		-R: random shuffle **-c**: check, if sorted, do not sort **-t**: field separator **-k**<*keydef*> : sort according to keydef <*keydef*>: F[.C][OPTS][,F[.C][OPTS]] **-u**: output only the first of equal lines
split	split a file into pieces	
tail	Output the last part of a file	**-n***<num>: print the last <num> lines* **-n** + *<num>: print, starting from line <num>*
tr	Translate, squeeze, and/or delete characters	**-c**: use the complement of set1 **-d**: delete the characters in set1 **-s**: replace each sequence of a repeated character with a single character of last given set
uniq	report or omit repeated lines	**-c :** prefix lines by the number of occurrences **-d** : only print duplicate lines, one for each group **-i**: ignore case **-u** : only print unique lines
wc	print newline, word, and byte counts for each file	**-c**: print byte counts **-m**: print the character counts **-w**: print the word counts **-l**: print the newline counts

sed – stream editor for filtering and transfering text

Command	Description	Popular options
sed	stream editor for filtering and transforming text	**-n :** suppress automatic printing of pattern space **-f** <script-file> : scripts with commands tob e executed **-i** : edit inplace **-E, -r** : support extended regexp
		<address> <start-address>,<end-address> <start-address>, + <number-of-lines> <address> can be: • line-number (i.e. 1,5,7, …) • $ (represent last line of file) • regular-expression

Sed commands	Description
a *<text>*	append text
i *<text>*	insert text
c *<text>*	replace the selected lines with text
p	print
d	delete pattern space
s/regexp/replacement/	regexp-replace

sed-Examples:

- ```
  # delete line(s) containing Aachen (inplace)
  sed -i '/Aachen/ d' city.csv
  ```

- ```
  # insert ,Karlsruhe …‘ at line 2
  sed '2i Karlsruhe,D,"Baden Wuerttemberg",301452,49.0,6.8'
  city.csv
  ```
- ```
  # remove all script-sections
  sed -Ei '/<script>/,/<\/script>/d' jaccard.html
  ```
- ```
  # replace NULL ->\n
  sed -i 's/\bNULL\b/\\N/g' city.csv
  ```
- ```
  # print lines 5-10, 23, 56-71
  sed -n '5,10p;23p;56,71p' city.csv
  ```

awk – pattern scanning and processing language

Command	Description	Popular options
awk	pattern scanning and processing language	**-F** field separator **-f** <script-file> : scripts with commands to be executed **-v**<key>=<value>: passing parameter <key> to script

awk-Examples:

- ```
  awk -F: '{printf  "%05d\t%s\t%s", $1, $3, $2}' input-file.txt
  ```
- ```
  awk -F' ' -f distribute-input-based-on-rules.awk input-file.txt
  ```

Version: 10.5.2018 - andreas.schmidt@kit.edu
  ```