

WPF - Information Retrieval

Wintersemester 2022, Andreas Schmidt

Coreutils Command Overview

Command	Description	Popular options
cat	concatenate files and print on the standard output	-n : number all output lines
head	output the first part of files	-n<num> : print the first <num> lines -n -<num> : print all but the last <num> lines
tail		-n<num> : print the last <num> lines -n + <num> : print, starting from line <num>
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
grep	print lines matching a pattern	-E : support extended regexp -i : ignore case -v : invert match -o : print only the matched part (output: one line per match) -f : obtain patterns from file -l : files with match -L : files without match -A <num> : Print <num> lines after matching context. -B <num> : Print <num> lines before matching context. -m<num> : Stop reading a file after <num> matching lines. -c : suppress normal output, instead count matching lines.
seq	print a sequence of numbers	-s : separator (default: \n)
split	split a file into pieces	-l <n> : put NUMBER lines/records per output file -t <sep> : use SEP instead of newline as the record separator
csplit	Output pieces of FILE separated by PATTERN(s) to files	
cut	remove sections from each line of files	-d<delim> : Use <delim> instead of ab as field separator -f<field-list> : select only fields in <field-list> --output-delimiter=<delim> : use <delim> as output delimiter
paste	merge lines of files	-d<char> : use <char> as output delimiter
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
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
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> : filename.field[[filename.field][...]] -a<filename>: Outer join semantic
Command	Description	Popular options
comm	compare two sorted files line by line	-1: suppress column 1 (lines unique to FILE1) -2: suppress column 1 (lines unique to FILE2) -3: suppress column 1 (lines unique to FILE3) --total : output a summary
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 -s <n>: avoid comparing the first <n> characters -f <n>: avoid comparing the first <n> fields

sed – stream editor for filtering and transferring 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 to be executed -i : edit inplace -E, -r : support extended regexp
		<address> <start-address>,<end-address> <start-address>, + <number-of-lines> <address> can be: <ul style="list-style-type: none"> • 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:

- `sed -i '/Aachen/ d' city.csv` # delete line(s) containing Aachen (inplace)
- `sed '2i Karlsruhe,D,"Baden Wuerttemberg",301452,49.0,6.8' city.csv`
insert ',Karlsruhe ...' at line 2
- `sed -Ei '/<script>/,/</script>/d' jaccard.html` # remove all script-sections
- `sed -i 's/\bNULL\b/\N/g' city.csv` # replace NULL ->\n (inplace)
- `sed -n '5,10p;23p;56,71p' city.csv` # print lines 5-10, 23, 56-71
- `sed '1,100d' city.csv` # delete line 1 - 100

awk-Examples:

- `awk -F, '$3=="Bavaria" && $4 < 1000000 { print $1,$4 }' city.csv`
- `awk 'rand() < 0.01 { print $0 }' big-file.tsv`
- write output to files corresponding to column 2
`awk -F$'\t' '{gsub("[^A-Za-z]", "", $2); print $1,$3 \`
`>> "data/"$2".txt"} ' life-expectancy.txt`
- Change record-separator from comma to tab:
 - File `csv2tsv.awk`:

```
BEGIN{
    FPAT = "([^\,]*)|(\"[^\"]*\")"
    OFS="\t"
}
{
    $1=$1 # (trigger awk to reconstruct output)
    print $0
}
```
- `awk -F, -f csv2tsv.awk bchr.csv > bchr.tsv`

Separator specification for different commands:

Command	Specification parameter	Default separator
cut	-d	<Tab>
awk	-F	<Blanc>, <Tab>
sort	-t	<Blanc>
Join	-t	<Blanc>
uniq		<Blanc>,<Tab>

Specification of <tab> (\t) as separator: `$'\t'` instead of `'\t'`.