Unix has tools for text processing from the very beginning (1970s)
- Small, simple tools, each tool doing only one operation
- Pipe (pipeline): powerful mechanism to combine tools
Short Description of Basic Text Tools

- **cat**: concatenate files and print on the standard output
- **head**: output the first part (few lines) of files
- **tail**: output the last part (few lines) of files
- **sort**: sort lines of text files
- **uniq**: remove duplicate lines from a sorted file
- **comm**: compare two sorted files line by line
- **wc**: print the number of newlines, words, and bytes in files
- **cut**: remove sections (columns) from each line of files
- **join**: join lines of two files on a common field
- **paste**: merge lines of files
- **tr**: translate or delete characters
Short Description of Basic Text Tools

egrep  prints lines matching a pattern
(g)awk  pattern scanning and processing language
sed  stream editor, use for substring replacement
use perl -p for extended regular expressions
info run info and select from a menu or run directly:
  ■ info coreutils
  ■ info head, info sort, ...
  ■ info gawk

man
  ■ man 7 regex
  ■ man grep, man awk, man tail, ...

–help most tools display a short help message on the
  --help option
  ■ sort --help, uniq --help, ...
Unix Text Tools Packages

Where to find it

- set of system tools
- different sets and different features/options on each Unix type
- GNU textutils
- GNU coreutils – textutils + shellutils + fileutils
- other GNU packages: grep, sed, gawk
set of system tools
- different sets and different features/options on each Unix type
- GNU textutils
- GNU coreutils – textutils + shellutils + fileutils
- other GNU packages: grep, sed, gawk
- installed on all Linux machines
- on Windows: install mingw32/cygwin, then coreutils, grep, ...
command line tools – enter command in a terminal (console) window
command name followed by options and arguments
options start with -
quote spaces and metacharacters: ’, ”, $
redirect input and output from/to files using <, >
use less to only display a result without saving
Text Tools Example 1

**task** Convert plain text file to a vertical text.

**input** plain.txt

**output** plain.vert

**solutions**

- tr -s ' ' '
' <plain.txt >plain.vert
- tr -sc a-zA-Z0-9 '
' <plain.txt >plain.vert
- perl -ne 'print "$&
" while /\b\w+|\[!\w\s]+\b/g' plain.txt >plain.vert
Text Tools Example 1

task  Convert plain text file to a vertical text.
input  plain.txt
output  plain.vert
solutions

tr -s ' ' '' \n' <plain.txt >plain.vert
task  Convert plain text file to a vertical text.
input  plain.txt
output  plain.vert
solutions

tr -s " " "\n" <plain.txt >plain.vert

tr -sc a-zA-Z0-9 "\n" <plain.txt >plain.vert
task  Convert plain text file to a vertical text.
input  plain.txt
output  plain.vert

solutions

```
tr -s ' ' '
' <plain.txt >plain.vert

tr -sc a-zA-Z0-9 '\n' <plain.txt >plain.vert

perl -ne 'print "$&\n" while /(^\|[^\w\s]+)/g' \ plain.txt >plain.vert
```
Task: Create a word list

Input: vertical text

Output: list of all unique words with frequencies

Solutions:

```
sort plain.vert | uniq -c >dict
sort plain.vert | uniq -c | sort -rn | head -10
```
Text Tools Example 2

task  Create a word list
input  vertical text
output  list of all unique words with frequencies
solutions

sort plain.vert | uniq -c >dict
sort plain.vert | uniq -c | sort -rn | head -10
Text Tools Example 3

task  Corpus/list size
  input  vertical text/word list
  output  number of tokens/different words
solutions
Text Tools Example 3

**task**  Corpus/list size

**input**  vertical text/word list

**output**  number of tokens/different words

**solutions**

```
wc -l plain.vert
wc -l dict
grep -c -i '^[a-z0-9]*$' plain.vert
```
Text Tools Example 4

Task: Create a list of bigrams
Input: vertical text
Output: list of bigrams
Solution:

```
tail +2 plain.vert | paste - plain.vert | sort | uniq -c > bigram
```
Text Tools Example 4

task  Create a list of bigrams
input  vertical text
output list of bigrams
solution

tail +2 plain.vert |paste - plain.vert \  
|sort |uniq -c >bigram
task  Filtering
input  word list
output  selected values from word list
solutions
Text Tools Example 5

task  Filtering
input  word list
output  selected values from word list
solutions

grep '^\[[0-9]*$' dict
awk '$1 > 100' dict
Text Tools Debugging

- data driven programming
- cut the pipeline and display partial results
- try single command with a test input
task  Find all words from a word list differing with s/z alternation only: apologize/apologise
task  Find all words from a word list differing with s/z alternation only: apologize/apologise

solutions

```
tr s z < dict | sort | uniq -d > szaltern
```
Find all words from a word list differing with s/z alternation only, and each alternation has higher frequency than 50
Text Tools Exercises

- Find all words from a word list differing with s/z alternation only, and each alternation has higher frequency than 50
- and display their frequencies
Text Tools Exercises

- Find all words from a word list differing with s/z alternation only, and each alternation has higher frequency than 50 and display their frequencies.
- Find all words which occurs in the word list only with capital letter (names).
Summary

Use simple Unix text tools for processing text files and computation of global statistics.

Use a powerful graphical user interface for local corpus exploration:
- Sketch Engine: www.sketchengine.co.uk
- Manatee/Bonito: www.textforge.cz
Use simple **Unix text tools** for processing text files and computation of **global** statistics.
Summary

- Use simple Unix text tools for processing text files and computation of global statistics.
- Use a powerful graphical user interface for local corpus exploration:
Summary

- Use simple **Unix text tools** for processing text files and computation of **global statistics**.
- Use a powerful **graphical user interface** for local corpus exploration:
  - Sketch Engine: www.sketchengine.co.uk
Use simple **Unix text tools** for processing text files and computation of **global** statistics.

Use a powerful **graphical user interface** for local corpus exploration:

- Sketch Engine: www.sketchengine.co.uk
- Manatee/Bonito: www.textforge.cz