Data Processing: Formats and Tools (part 2)

a topic in

DM565 – Formal Languages and Data Processing

Kim Skak Larsen

Department of Mathematics and Computer Science (IMADA)
University of Southern Denmark (SDU)

kslarsen@imada.sdu.dk

September, 2019

Some Relatively Simple Command-Line Tools

- sort
- uniq
- tr
- cut
- paste
- join
- head/tail

sort

Options for common issues (selected)

- ignore blanks
- ignore case
- sort numerically, alphabetically, by month, version numbers, . . .
- specify which field to sort on
- specify delimiters
- reverse

uniq

"Filter out adjacent matching lines" – often used after `sort`

Options for common issues (selected)

- ignore case
- print only unique or duplicate lines
- consider only the first or last some number of characters
- consider only some fields
- count duplicates

tr

"translate or delete characters"

Options for common issues (selected)

- delete characters in a given set
- delete consecutive duplicates of a character, leaving one occurrence
- translate by specifying two character sequences of the same length

cut

"remove sections from each line of files"

Options for common issues (selected)

- select numbered bytes
- only keep certain characters
- select some fields
- specify delimiter
- specify output delimiter

paste

"merge lines of files"

Options for common issues (selected)

- specify delimiter (default is `\t`)
- serial mode (each file will be a line)

join

“join lines of two files on a common field” – similar to dbms equi-join

Options for common issues (selected)

- ignore case
- specify delimiters
- specify join field

head/tail

“output the first/last part of files”

Options for common issues (selected)

- specify the number of lines
- specify bytes instead of lines

Command-Line Tools: sed and awk

We will use the lecture notes from New York University:

`https://cs.nyu.edu/~mohri/unix08/lect5.pdf`

JSON-Like Formats: JSON

“JavaScript Object Notation”

JSON

```
{ "animals": [
    {
      "name": "Panda",
      "cuteness": 1.0,
      "colors": [ "white", "black" ]
    },
    {
      "name": "Panther",
      "cuteness": 0.7,
      "colors": [ "black" ]
    }
  ]
}
```

JSON-Like Formats: XML

"eXtensible Markup Language"

XML

```
<animals>
  <animal>
    <name>Panda</name>
    <cuteness>1.0</cuteness>
    <color>white</color>
    <color>black</color>
  </animal>
  <animal>
    <name>Panther</name>
    <cuteness>0.7</cuteness>
    <color>black</color>
  </animal>
</animals>
```

JSON-Like Formats

- There is more to both formats.
- The essence is that it is named parentheses structures expressing records (attribute/value pairs) and sequences (arrays, lists).
- There are many variants of XML (HTML) with similar structure.
- Command-Line tools can to some extent be used for data discovery, and possibly simpel code execution.
- To get full power, use a programming language with an appropriate package.
- Packages read json/xml files and deliver data in native formats.

JSON-Like Formats: Python Example

```
> cat animals.json
{ "animals": [
    {
      "name": "Panda",
      "cuteness": 1.0,
      "colors": [ "white", "black" ]
    },
    {
      "name": "Panther",
      "cuteness": 0.7,
      "colors": [ "black" ]
    }
  ]
}
>
```

JSON-Like Formats: Python Example

```
# Prints
# {
#     "a": 1,
#     "b": 2,
#     "c": 3,
#     "d": 4,
#     "e": 5
# }
# Panda

import json

# Data in program for testing
json_data = '{"a": 1, "b": 2, "c": 3, "d": 4, "e": 5}'

parsed_json = (json.loads(json_data))
print(json.dumps(parsed_json, indent=4, sort_keys=True))

# It is just dictionaries and lists
with open('animals.json', 'r') as f:
    animals_dict = json.load(f)
print(animals_dict["animals"][0]["name"])
```