

Текстовая расшифровка видео:

ПОТОКОВЫЕ РЕДАКТОРЫ И ФОРМАТЫ

План:

- Поточковый редактор sed;
- Поточковый редактор awk;
- Работа с CSV;
- Работа с JSON;
- Пример реального пайплайна.

Поточковый редактор sed

Потоковых редакторов достаточно много. Самые популярные:

- Sed;
- Awk.

Sed – это практически маленький язык программирования, который многое умеет.

Чаще всего в задачах дата-инженеринга или дата-сайенса sed используют для замены строк в строках.

Это может выглядеть следующим образом:

```
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/zed/'
zed foo omg bar lol
meow-nofer@pikachu ~/Experiments/slurm
```

(с 0:51)



Например, у нас есть какая-либо строка, и мы говорим «**sed 's'**» (оператор, которого мы применяем). Через слэш задаем первый аргумент, второй аргумент – на что его меняем. Если мы выполним эту команду, то заметим следующее: замена отработала только для первого вхождения в строку. Чаще всего мы хотим заменить слово во всей строке. В конце добавляем модификатор «g». Так, у нас заменятся все слова:

```
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/zed/'
zed foo omg bar lol
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/zed/g'
zed foo omg zed lol
meow-nofer@pikachu ~/Experiments/slurm
```

В sed есть интересный синтаксис. Например, можно сослаться во втором аргументе на то, что мы выбрали в первом аргументе.

Например, мы хотим взять все слова в скобки. Можно сослаться на все амперсандом:

```
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/zed/'
zed foo omg bar lol
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/zed/g'
zed foo omg zed lol
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/(&)/g'
(bar) foo omg (bar) lol
meow-nofer@pikachu ~/Experiments/slurm
```

Вопрос: что делать в ситуации, когда у нас путь в файловой системе?

Ответ: sed в качестве разделителя между операцией, операндами и модификаторами может использовать любой символ, который не присутствует внутри выражения:

```
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/zed/'
zed foo omg bar lol
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/zed/g'
zed foo omg zed lol
meow-nofer@pikachu ~/Experiments/slurm echo "bar foo omg bar lol" | sed 's/bar/(&)/g'
(bar) foo omg (bar) lol
meow-nofer@pikachu ~/Experiments/slurm echo "/a/b/c" | sed 's/\/a\/b\/d\/e/g'
/d/e/c
meow-nofer@pikachu ~/Experiments/slurm echo "/a/b/c" | sed 's:/a/b:/d/e:g'
/d/e/c
meow-nofer@pikachu ~/Experiments/slurm
```

Также можно использовать расширенные синтаксисы регулярных выражений и т.д. Например, у нас есть группа «foo». Мы ссылаемся на нее и заменяем все слова на другие. **С заменами можно делать все, что угодно.**

Говоря о **tr**, стоит сказать, что **он заменяет посимвольно.**

Предлагаем ознакомиться с глобальным tutorialом по sed:

<http://www.grymoire.com/Unix/Sed.html>

Потоковый редактор awk

Основное и базовое предназначение **awk** – разделение входных данных на колонки.

Когда мы работаем с файлами или процессами, у нас есть команда «**ls - la**», где все данные выводятся в колоночном формате:

csvkit — модуль Python для продвинутой работы с CSV

```
~$ pip install csvkit
```

in2csv, csvcut, csvlook, csvjson, csvsql, csvsort

```
~$ in2csv imdb-250-1996-2011-lists-only.xlsx 2>/dev/null | csvsql --query "select Title,Year from stdin where Year<2009" | csvsort -r -c Year | head -n 10 | csvlook
```

Среди команд csvkit есть команда **in2csv**, которая работает с разными форматами. Например, в качестве источника данных выступает файл Excel с топ-250 фильмов с 1996-2011 годы.

Установим csvkit:

```
meow-nofer@pikachu ~/Experiments/slurm ls
76-0.txt 76-ru.txt 76-ru-utf.txt airport-codes.csv imdb-250-1996-2011-lists-only.xlsx
meow-nofer@pikachu ~/Experiments/slurm mkvirtualenv 3.10.6 test_csv
pyenv-virtualenv: `3.10.6' is not installed in pyenv.
pyenv-virtualenv: version `test_csv' is not a virtualenv
X meow-nofer@pikachu ~/Experiments/slurm lsvirtualen
```

(с 13:21)

```
3.10.9/envs/geo
3.10.9/envs/geodev
3.10.9/envs/jup
3.11.1
aioes --> /home/meow-nofer/.pyenv/versions/3.10.5/envs/aioes
ans --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/ans
bots --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/bots
cookie --> /home/meow-nofer/.pyenv/versions/3.10.5/envs/cookie
dbtenv --> /home/meow-nofer/.pyenv/versions/3.9.15/envs/dbtenv
debugpy --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/debugpy
dj --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/dj
fake --> /home/meow-nofer/.pyenv/versions/3.10.5/envs/fake
fast --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/fast
geo --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/geo
geodev --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/geodev
hub --> /home/meow-nofer/.pyenv/versions/3.10.5/envs/hub
i3 --> /home/meow-nofer/.pyenv/versions/3.10.5/envs/i3
jup --> /home/meow-nofer/.pyenv/versions/3.10.9/envs/jup
neovim2 --> /home/meow-nofer/.pyenv/versions/2.7.15/envs/neovim2
neovim3
pipe --> /home/meow-nofer/.pyenv/versions/3.10.5/envs/pipe
slurm_3 --> /home/meow-nofer/.pyenv/versions/3.10.5/envs/slurm_3
smtp --> /home/meow-nofer/.pyenv/versions/3.8.10/envs/sntp
```

```
Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting sqlalchemy<2
  Downloading SQLAlchemy-1.4.48-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x
64.whl (1.6 MB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 6.6 MB/s eta 0:00:00
Collecting et-xmlfile
  Using cached et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Collecting text-unidecode>=1.3
  Using cached text_unidecode-1.3-py2.py3-none-any.whl (78 kB)
Collecting greenlet!=0.4.17
  Using cached greenlet-2.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (613 kB)
```

```
Installing collected packages: text-unidecode, pytimeparse, parsedatetime, dbfread, xlrd, six, python-slugify, olefile, gre
enlet, et-xmlfile, Babel, sqlalchemy, openpyxl, leather, isodate, agate, agate-sql, agate-excel, agate-dbf, csvkit
DEPRECATION: olefile is being installed using the legacy 'setup.py install' method, because it does not have a 'pyproject
.toml' and the 'wheel' package is not installed. pip 23.1 will enforce this behaviour change. A possible replacement is to
enable the '--use-pep517' option. Discussion can be found at https://github.com/pypa/pip/issues/8559
Running setup.py install for olefile ... done
Successfully installed Babel-2.12.1 agate-1.7.1 agate-dbf-0.2.2 agate-excel-0.2.5 agate-sql-0.5.9 csvkit-1.1.1 dbfread-2.0.
7 et-xmlfile-1.1.0 greenlet-2.0.2 isodate-0.6.1 leather-0.3.4 olefile-0.46 openpyxl-3.1.2 parsedatetime-2.6 python-slugify-
8.0.1 pytimeparse-1.1.8 six-1.16.0 sqlalchemy-1.4.48 text-unidecode-1.3 xlrd-2.0.1

[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: python3.10 -m pip install --upgrade pip
(test_csv) meow-nofer@pikachu ~/Experiments/slurm in2csv imdb-250-1996-2011-lists-only.xlsx 2>/dev/null
[0] 0:zsh* "pikachu" 14:01 29-мая-23
```

Запустим команду. Мы видим, что файл Excel распарсился, тем самым сгенерировался валидный csv:

```
Fight Club (1999),FightClub(1999),1999,14,237,8.8,,18,4,
The Matrix (1999),TheMatrix(1999),1999,21,230,8.7,,27,6,
The Green Mile (1999),TheGreenMile(1999),1999,74,177,8.4,,85,11,
Memento (2000),Memento(2000),2000,31,220,8.6,,29,-2,
Requiem for a Dream (2000),RequiemforaDream(2000),2000,61,190,8.4,,61,0,
Amores Perros (2000),AmoresPerros(2000),2000,164,87,8.1,,164,0,
Snatch. (2000),Snatch.(2000),2000,124,127,8.2,,130,6,
Gladiator (2000),Gladiator(2000),2000,88,163,8.3,,96,8,
Donnie Darko (2001),DonnieDarko(2001),2001,145,106,8.2,,128,-17,
"Monsters, Inc. (2001)", "Monsters,Inc.(2001)",2001,245,6,8,,239,-6,
```

```
"Fabuleux destin d'Amélie Poulain, Le (2001)", "Fabuleuxdestind'AméliePoulain,Le(2001)",2001,49,202,8.5,,47,-2,
The Lord of the Rings: The Fellowship of the Ring(2001),TheLordoftheRings:TheFellowshipoftheRing(2001),2001,17,234,8.7,,20,
3,
Sen to Chihiro no kamikakushi(2001),SentoChihironokamikakushi(2001),2001,45,206,8.5,,54,9,
The Lord of the Rings: The Two Towers (2002),TheLordoftheRings:TheTwoTowers(2002),2002,30,221,8.6,,31,1,
Cidade de Deus(2002),CidadedeDeus(2002),2002,18,233,8.7,,19,1,
The Pianist (2002),ThePianist(2002),2002,51,200,8.5,,53,2,
Mou gaan dou (2002),Mougaandou(2002),2002,228,23,8,,243,15,
Big Fish (2003),BigFish(2003),2003,227,24,8,,203,-24,
Finding Nemo (2003),FindingNemo(2003),2003,171,80,8.1,,154,-17,
Pirates of the Caribbean: The Curse of the Black Pearl (2003),PiratesoftheCaribbean:TheCurseoftheBlackPearl(2003),2003,232,
19,8,,219,-13,
Mystic River (2003),MysticRiver(2003),2003,222,29,8,,212,-10,
Kill Bill: Vol. 1 (2003),KillBill:Vol.1(2003),2003,142,109,8.2,,135,-7,
[0] 0:[tmux]* "pikachu" 14:01 29-мая-23
```

Давайте попробуем применить команду «**CSV SQL**». Мы выберем название и год для всех фильмов, которые вышли до 2009 года. Далее, есть возможность отсортировать их по году и посмотреть из них 10. Команда «**csvlook**» выведет все в форме красивой таблицы.

Посмотрим, как отработает пайплайн:

```
Up (2009),Up(2009),2009,102,149,8.3,,88,-14,
The Secret in Their Eyes (2009),TheSecretinTheirEyes(2009),2009,161,90,8.1,,169,8,
Toy Story 3 (2010),ToyStory3(2010),2010,38,213,8.5,,16,-22,
Inception (2010),Inception(2010),2010,11,240,8.8,,4,-7,
How to Train Your Dragon (2010),HowtoTrainYourDragon(2010),2010,182,69,8.1,,184,2,
(test_csv) meow-nofer@pikachu ~/Experiments/slurm in2csv imdb-250-1996-2011-lists-only.xlsx
very "select Title,Year from stdin where Year<2009" | csvsort -r -c Year | head -n 10 | csvlook
/home/meow-nofer/.pyenv/versions/3.10.9/envs/test_csv/lib/python3.10/site-packages/agatesql/table.
g: Deprecated API features detected! These feature(s) are not compatible with SQLAlchemy 2.0. To prevent incompatibility upgra
des prior to updating applications, ensure requirements files are pinned to "sqlalchemy<2.0". Set environment variable SQLA
LCHEMY_WARN_20=1 to show all deprecation warnings. Set environment variable SQLALCHEMY_SILENCE_UBER_WARNING=1 to silence t
his message. (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
| Title | Year |
| -----|-----|
| Ip Man (2008) | 2 008 |
| The Wrestler (2008) | 2 008 |
| Slumdog Millionaire (2008) | 2 008 |
| In Bruges (2008) | 2 008 |
| Låt den rätte komma in | 2 008 |
| Gran Torino (2008) | 2 008 |
| WALL·E (2008) | 2 008 |
| The Dark Knight (2008) | 2 008 |
| The Bourne Ultimatum (2007) | 2 007 |
(test_csv) meow-nofer@pikachu ~/Experiments/slurm
[0] 0:zsh* "pikachu" 14:02 29-мая-23
```

Также есть команда «**csvcut**», которая позволяет доставать колонки из исходного кода по названию.

Команда «**cut**» – еще одна команда извлечения колонки из файла, но на практике **awk** все равно быстрее.

Можно посмотреть на похожую утилиту – **Miller**. Она написана на GO и гораздо быстрее. Ее минус в том, что утилита написана не на Python.

Если посмотреть рейтинги программного обеспечения за всю историю, то на одном из первых мест будет старая программа **VisiCalc** – «дедушка» Excel.

Существует опенсорсная версия – **Visidata**.

Работа с JSON

JQ – это полноценный небольшой интерпретатор языка запросов и преобразований для JSON-документов.

Обратите внимание на примеры:

```
~$ sudo apt-get install jq

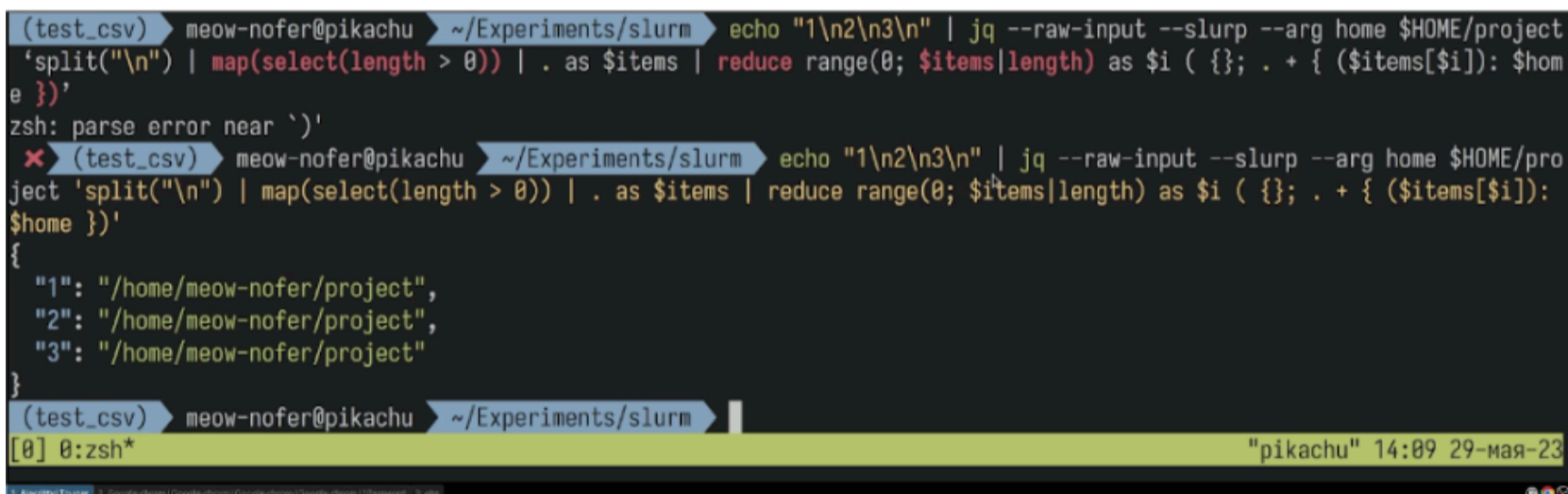
~$ cat file.txt | jq '.Title, .Year' # имеем дело с одним объектом

~$ cat file.txt | jq -c '[][.Title]' # имеем дело со списком

~$ cat file.txt | jq --raw-input --slurp --arg home $HOME/project 'split("\n") |
map(select(length > 0)) | . as $items | reduce range(0; $items|length) as $i ( {}; . + {
($items[$i]): $home } )'
```

Мы можем как доставать поля через точку объектов по имени, так и доставать целые списки и разворачивать их. Также мы можем строить JSON налету (нижний пример).

Давайте попробуем запустить:



```
(test_csv) meow-nofer@pikachu ~/Experiments/slurm echo "1\n2\n3\n" | jq --raw-input --slurp --arg home $HOME/project
'split("\n") | map(select(length > 0)) | . as $items | reduce range(0; $items|length) as $i ( {}; . + { ($items[$i]): $home
e } )'
zsh: parse error near `)'
✘ (test_csv) meow-nofer@pikachu ~/Experiments/slurm echo "1\n2\n3\n" | jq --raw-input --slurp --arg home $HOME/pro
ject 'split("\n") | map(select(length > 0)) | . as $items | reduce range(0; $items|length) as $i ( {}; . + { ($items[$i]):
$home } )'
{
  "1": "/home/meow-nofer/project",
  "2": "/home/meow-nofer/project",
  "3": "/home/meow-nofer/project"
}
(test_csv) meow-nofer@pikachu ~/Experiments/slurm
[0] 0:zsh* "pikachu" 14:09 29-мая-23
```

В консоли мы разобрали строчку, превратив ее в JSON.

Пример реального пайплайна

```
~$ cat file.txt | csvjson --stream | jq -c 'if .createdDate != "" then .createdDate =
(.standardRegCreatedDate | split(" ") | .[0:2] | join("T") + "Z" ) else .createdDate =
"9999-01-01T00:00:00Z" | to_entries | map(select(.key | contains("rawText") | not ) )
| from_entries'
```

Как вам урок?



Далее >