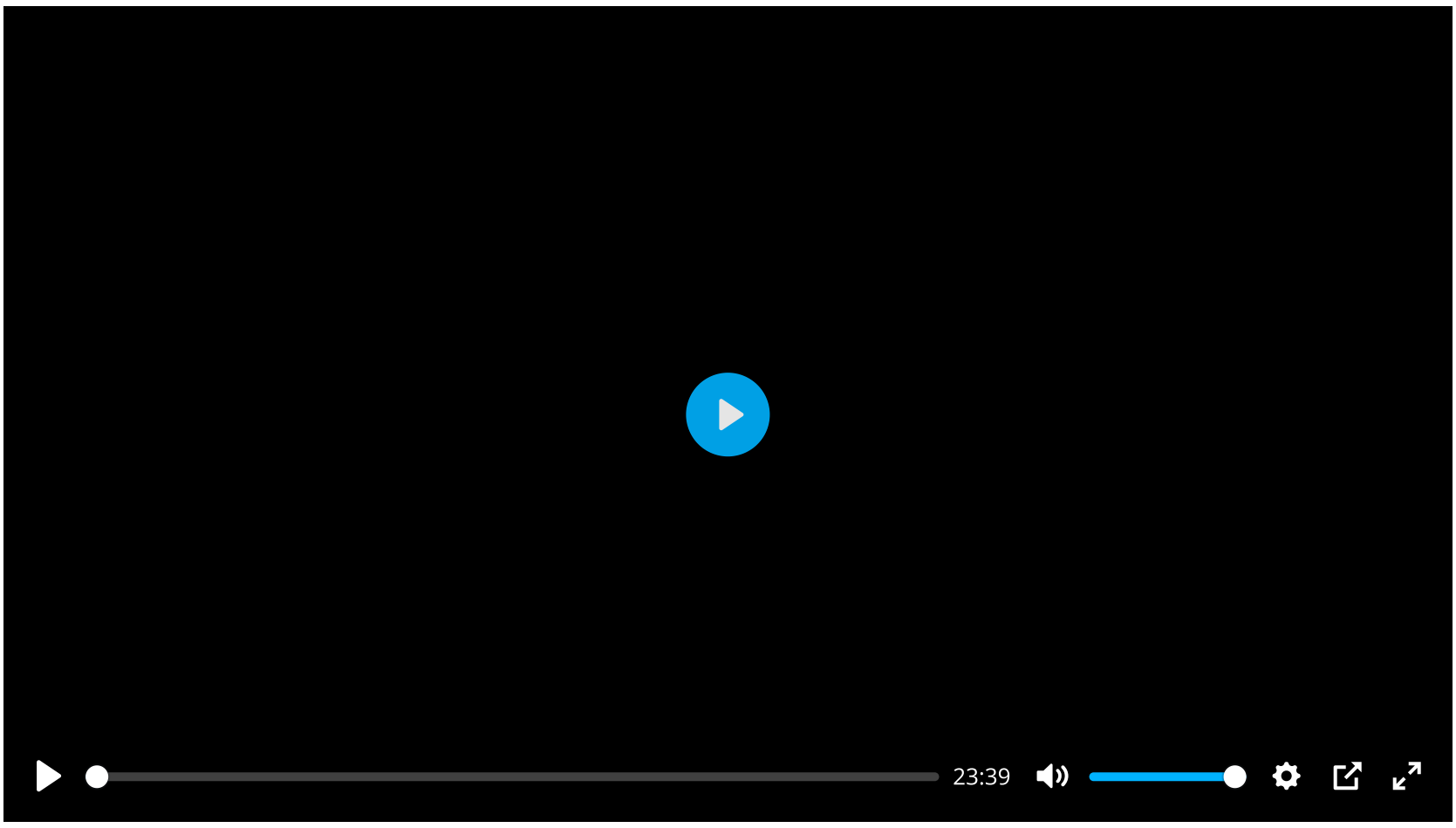
2.5 Linux терминал для задач DE



Текстовая расшифровка видео:

ПОТОКОВЫЕ РЕДАКТОРЫ И ФОРМАТЫ

План:

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

Потоковый редактор sed

Потоковых редакторов достаточно много. Самые популярные:

- Sed;
- Awk.

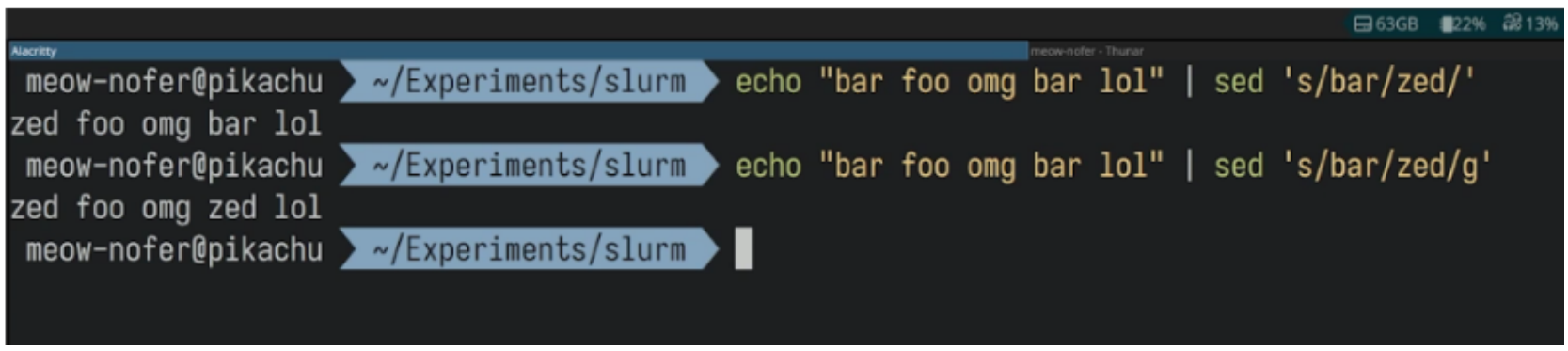
Sed – это практически маленький язык программирования, который многое умеет.

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

Это может выглядеть следующим образом:

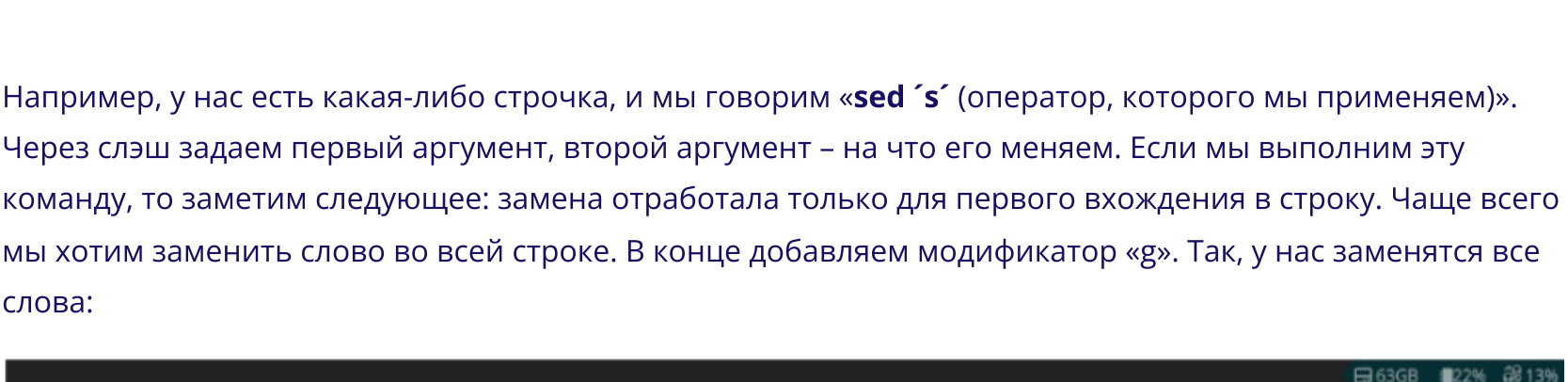
(с 0:51)

Например, у нас есть какая-либо строчка, 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:



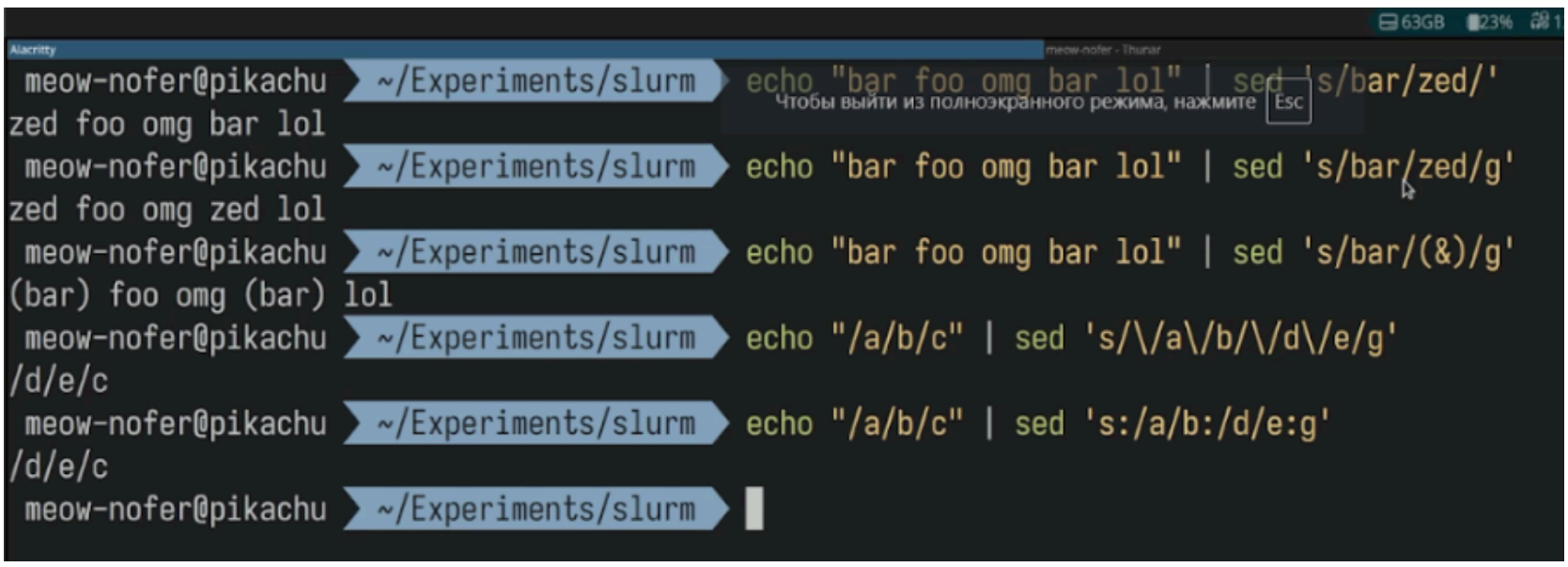
В sed есть интересный синтаксис. Например, можно сослаться во втором аргументе на то, что мы выбрали в первом аргументе.

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



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:



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 туториалом по sed:

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

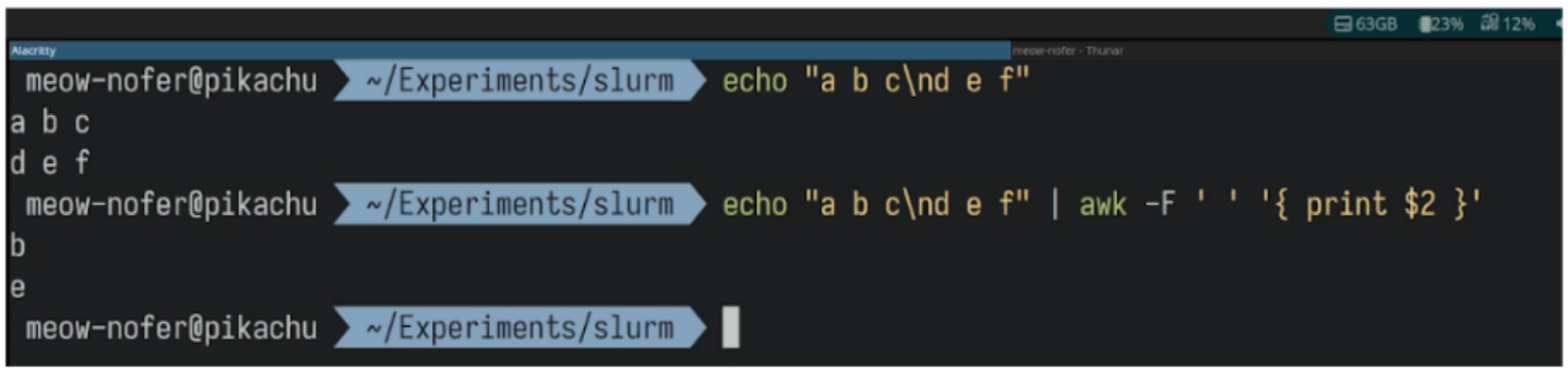
Потоковый редактор awk

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:

Банальное, что мы можем сделать с колонкой – вывести из нее значения:

Например, у нас есть файл с колонками **a, b, c, d, e, f.** Мы хотим выбрать только одну колонку. Мы можем сказать «awk выводи мне, пожалуйста, только колонку №2» (условно):



Запуск awk состоит из трех разных стадий:

- **BEGIN** – можем задавать стартовые значения;
- **Общая стадия** (без названия) – можем что-либо делать со строчками;
- **END** – можем выводить итоговый результат.

Внутри языка **awk не всегда доступно переменное окружение**. Это можно сделать при помощи ключа -v. Таким образом мы переносим переменные внутри awk и присваиваем значения, например, переменному внешнему окружению. Текущий shell разворачивает переменную, она подставляется в переменной самого awk.

В awk есть встроенные переменные, которые всегда можно использовать в коде:

- **NF** – число полей;
- **NR** – число строк.

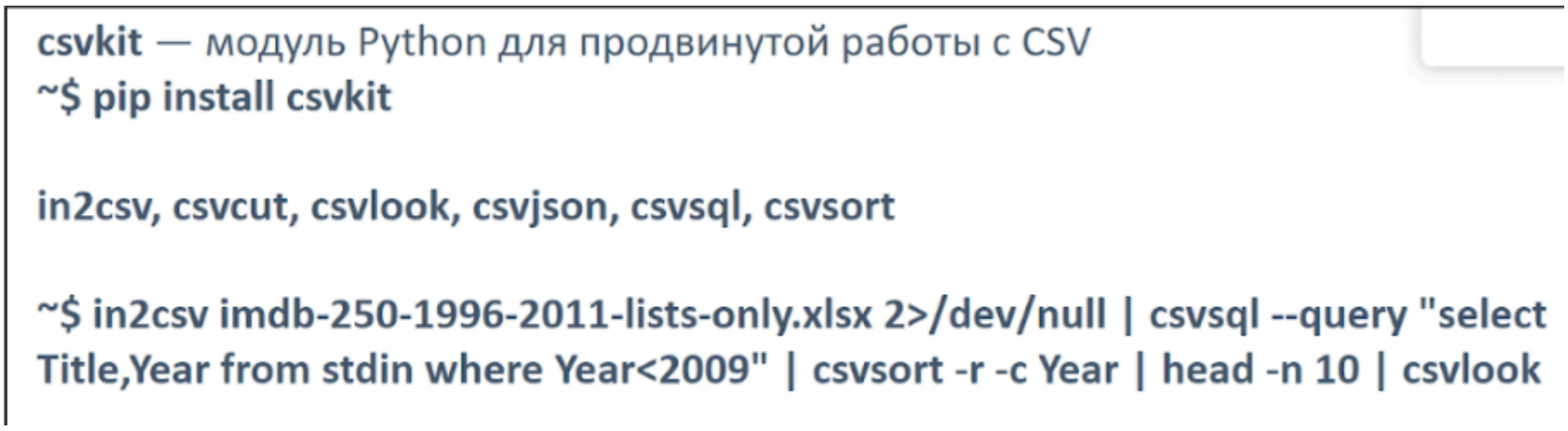
Предлагаем ознакомиться с глобальным туториалом по awk:

http://www.grymoire.com/Unix/Awk.html

Работа с CSV

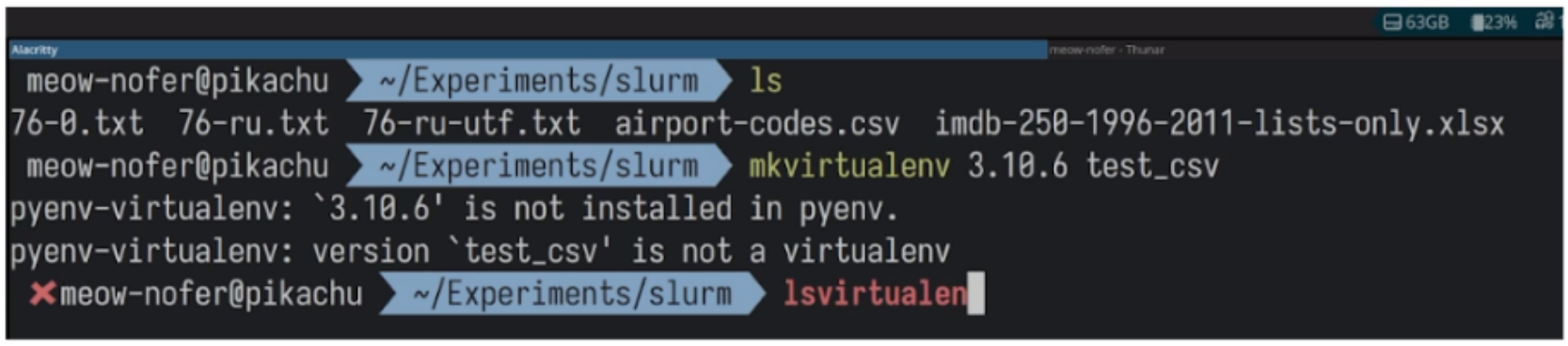
CSVkit – модуль Python для продвинутой работы с CSV.

Рассмотрим пример:

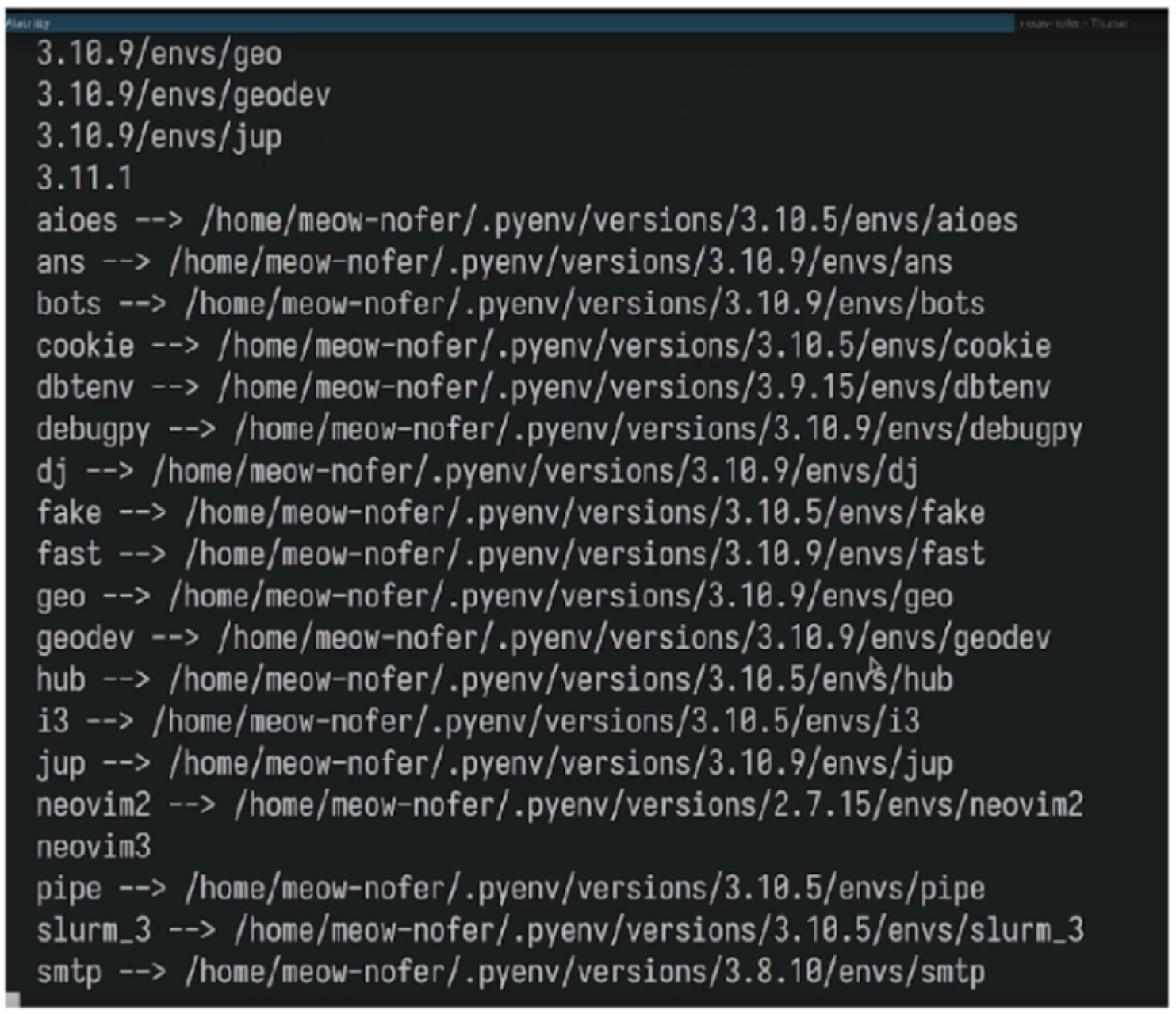


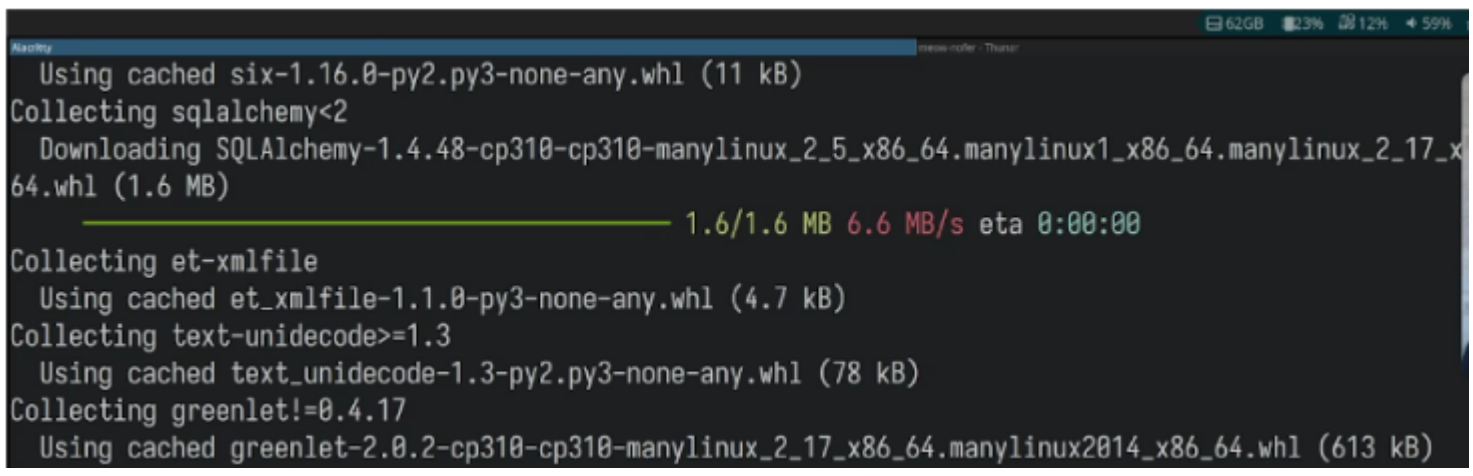
Среди команд csvkit есть команда **in2csv**, которая работает с разными форматами. Например, в качестве источника данных выступает файл Excel с топ-250 фильмов с 1996-2011 годы.

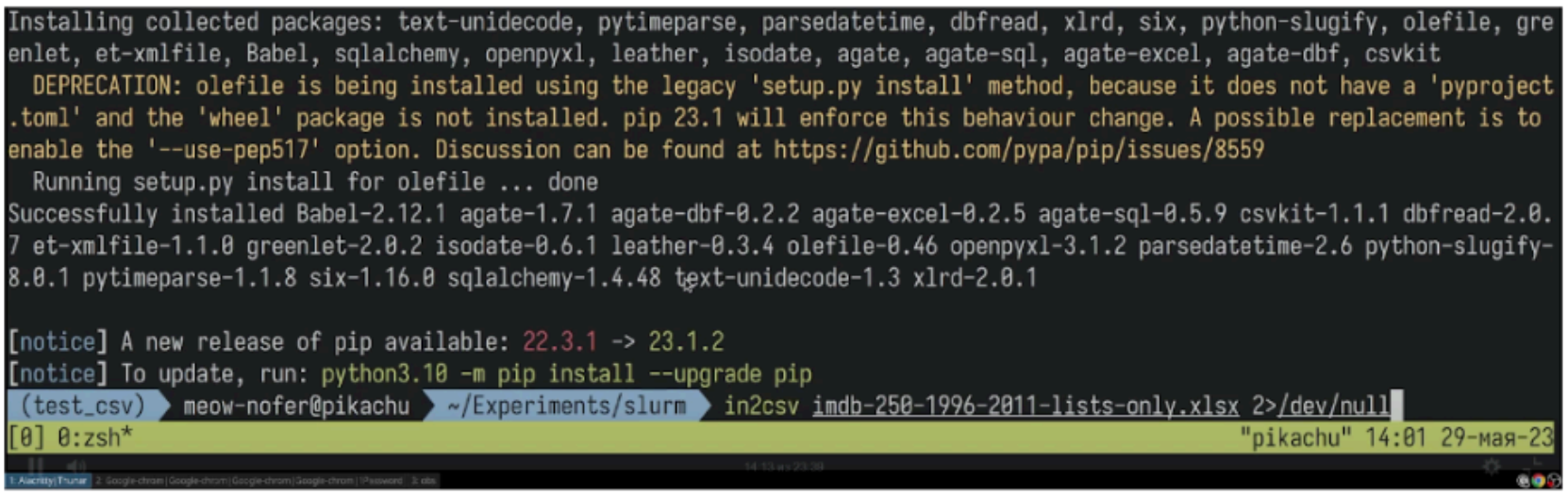
Установим csvkit:



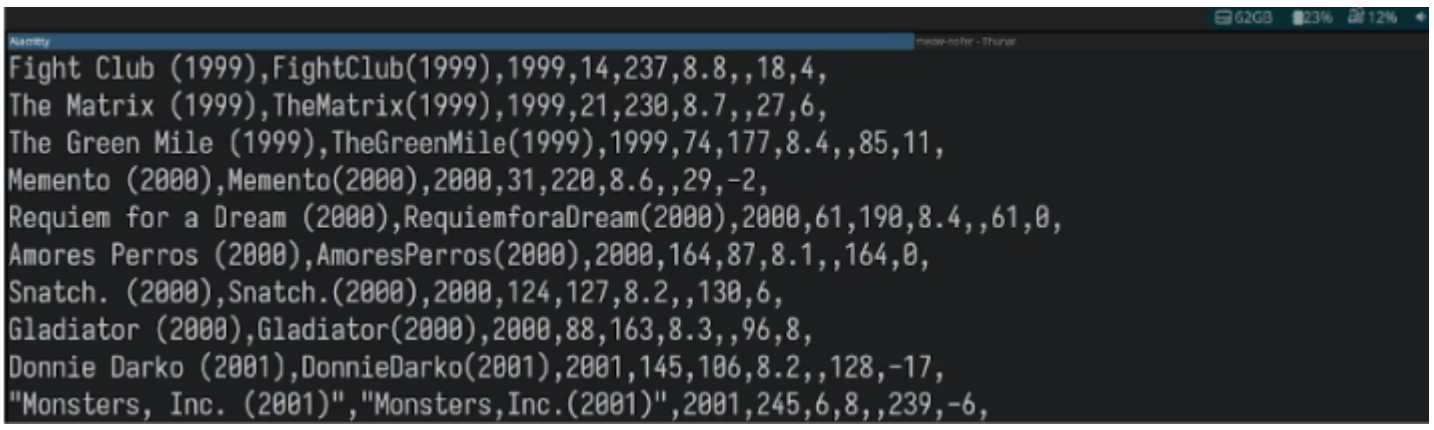
(c 13:21)

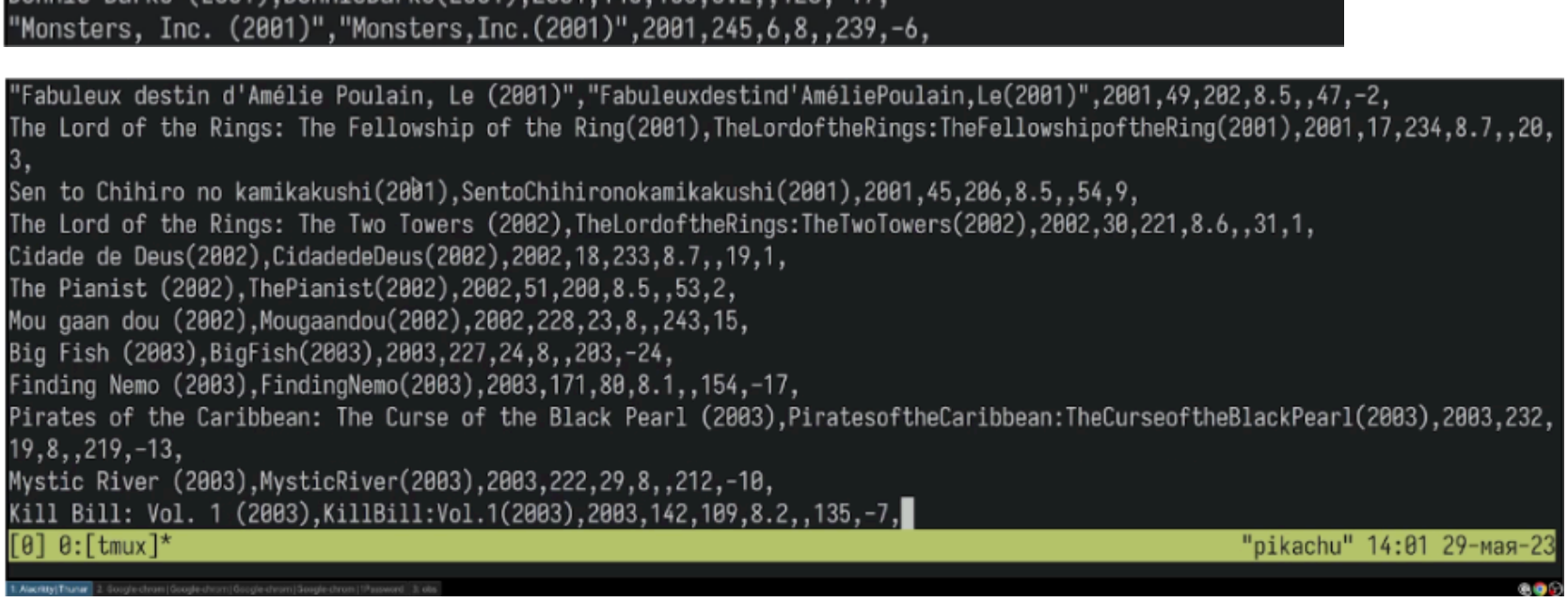






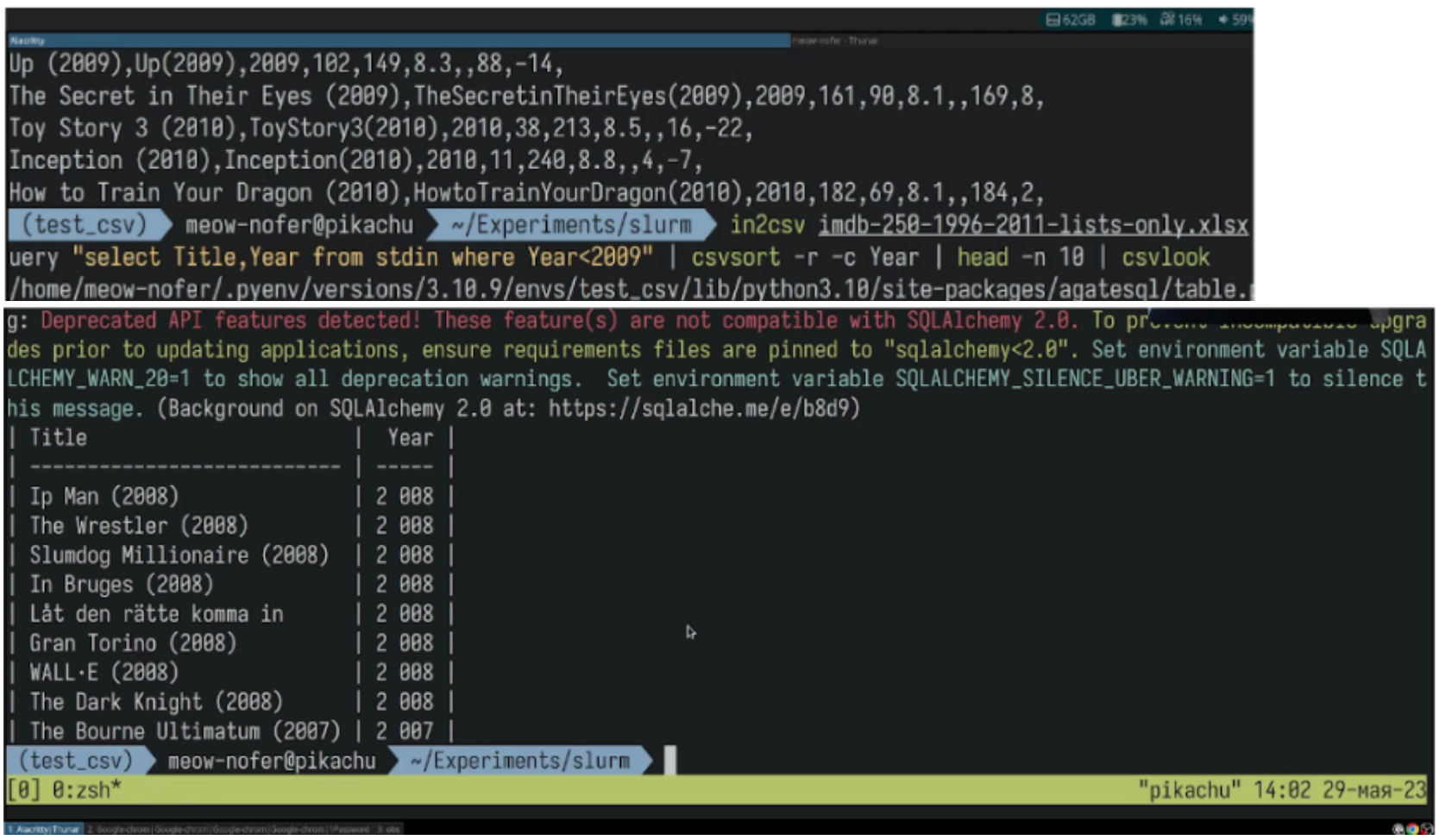
Запустим команду. Мы видим, что файл Excel распарсился, тем самым сгенерировался валидный csv:





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:



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

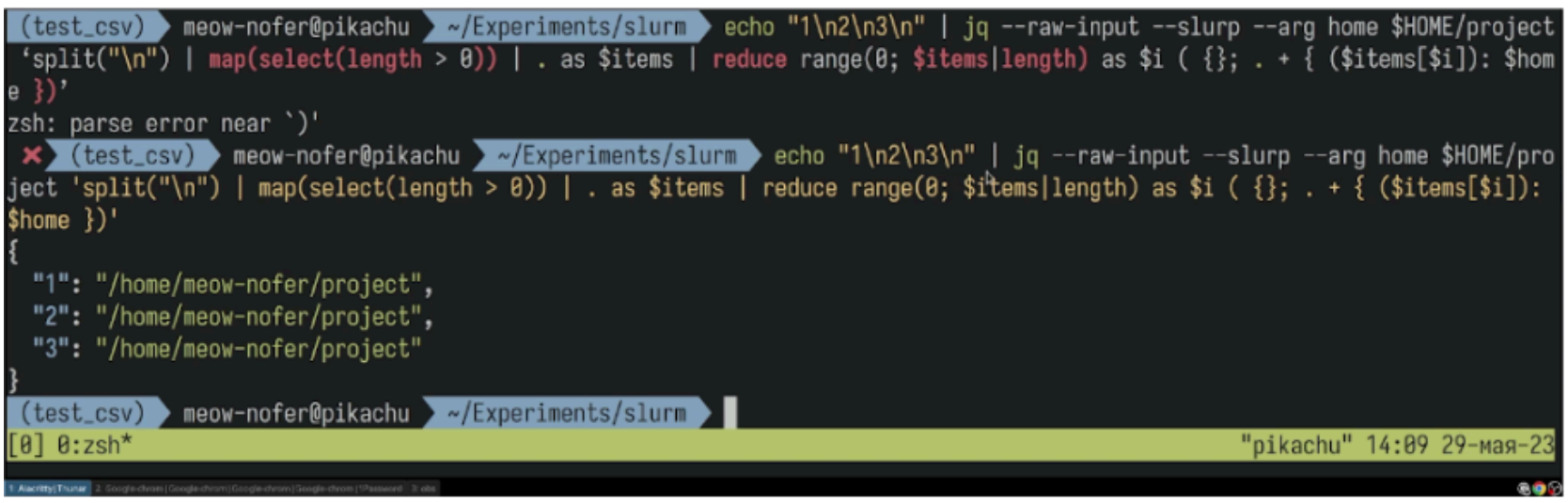
JQ – это полноценный небольшой интерпретатор языка запросов и преобразований для JSON-документов.

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:



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