Санкт-Петербургский государственный университет

Кафедра системного программирования

Утилита для парсинга SCSI логов

Кузнецов Арсений Николаевич, группа 22.Б15-мм

Научный руководитель: к. ф.-м. н., доцент кафедры СП, Куликов Е. К.
Консультант: Казаков И. А., старший инженер по разработке ПО ООО «Ядро Центр Программных Разработок»

Санкт-Петербург
2024

Проблема

- MeyerSAN — закрытый программно-аппаратный комплекс для тестирования СХД путем внедрения ошибок во взаимодействие СХД со SCSI[1] дисками.
- Для расширения MeyerSAN хотят использовать данные об ошибках произошедших с реальными дисками. Имеем в основном архивы с логами СХД, полученными через dmesg
- Планируется анализировать много информации по SCSI, но архивы логов, которые зачастую имеют сложную структуру и размер, исчисляемый в гигабайтах, далеко не самое удобное представление

[1] https://www.snia.org/education/online-dictionary/term/small-computer-system-interface

Существующие решения

- ELK Stack, fluent bit, rsyslog — подходят слабо, направлены скорее на сбор и взаимодействие с логами в целом, а не анализ в одной конкретной области
- «grep» по архивам — ugrep или AVFS + grep. При таком подходе может быть сложно искать конкретную информацию и в дальнейшем использовать машинные средства обработки

Предлагаемое решение

Утилита для парсинга Linux midlayer SCSI логов, которая может работать над архивами и выдавать разобранные логи в виде JSON. Так же утилита должна уметь фильтровать файлы в архиве по имени, чтобы не парсить не нужные файлы.
Преимущества подхода:

- Легко использовать
- JSON машиночитаем и писать какие-нибудь обработчики для результата будет намного легче
- Для анализа JSON есть множество утилит, например, jq

Постановка задачи

Целью работы является создание утилиты для анализа SCSI логов с возможностью работы над архивом и фильтрацией файлов в нем, а также сериализацией результата анализа в JSON.

Задачи:

- Спроектировать и реализовать модуль для парсинга SCSI логов, который может сериализовать результат в JSON
- Спроектировать и реализовать модуль для работы с архивами, который должен по архиву возвращать распакованные текстовые файлы с логами, удовлетворяющие предикату на имя
- Реализовать CLI с возможностью запуска функциональности перечисленных выше модулей

Парсинг логов

Поскольку строку лога можно разделить на несколько частей, где каждая следующая часть зависит от предыдущей, было решено реализовать парсер в виде дерева, состоящего из парсеров отдельных частей строки.
Результатом работы каждого парсера является `LogUnit`. А всего дерева парсеров в целом — интрузивный список из LogUnit, который можно сериализовать в JSON.

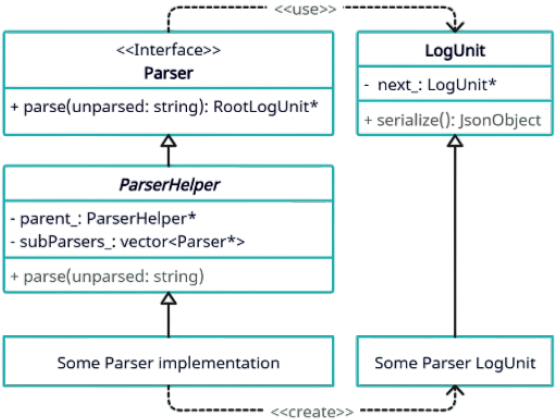
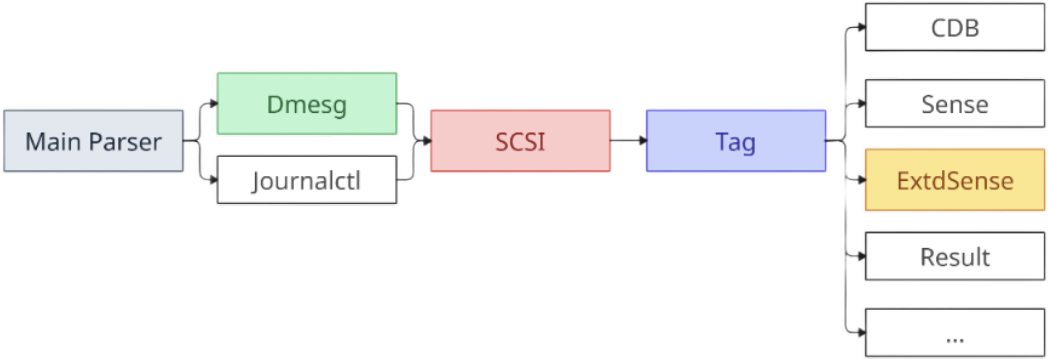


Рис.: Упрощенная UML абстракций парсера

Пример парсинга строки

kern :info : [Sat Mar 9 13:38:21 2024] sd 1:0:102:0: [sdkg] tag#5 Add. Sense: Unrecovered read error



Работа с архивами

- Задача: нужно получить текстовые данные по архиву, валидируя их по именам
- В этом поможет библиотека `libarchive`, которая может распаковывать данные в нужных форматах
- Реализация:
  - Лениво проходимся по элементам архива
  - Если встретился текстовый файл, то проверяем его имя предикатом
  - Если встретился сжатый файл — распаковываем
  - Если архив, то распаковываем его и рекурсивно проходимся над данными в нем
- Так же компонента, отвечающая за работу с архивами, может работать с папками и текстовыми файлами

CLI

- Реализован на основе библиотеки `CLI11`
- Имеет подкоманды `source` и `parse`
- `source` — отвечает за источник логов и `glob` предикат на имена
- `parse` — запуск парсера и параметры вывода результатов и сериализации
- Пример использования:
  `unilogs source archive.tar.gz parse -p --output 'result.json'`

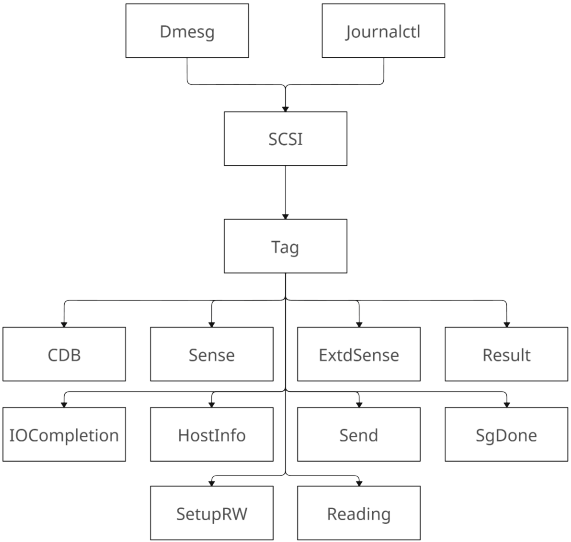
Результаты

- Реализована библиотека для парсинга SCSI логов, результат парсинга которой может быть сериализовать в JSON
- Реализована библиотека для работы с архивами, файлами, папками, которая может возвращать по ним текстовые файлы, удовлетворяющие предикату
- Реализован CLI

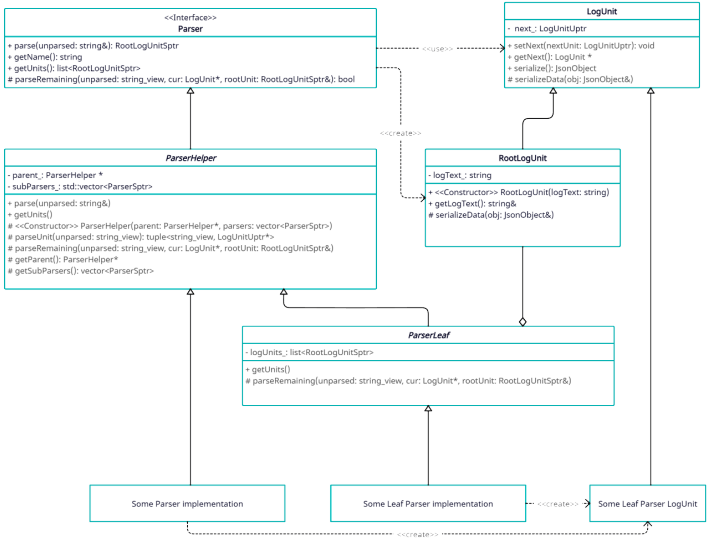
Ссылки

- ELK Stack — https://www.elastic.co/elastic-stack
- fluent bit — https://www.fluentbit.io/
- rsyslog — https://www.rsyslog.com/doc/index.html
- ugrep — https://github.com/Genivia/ugrep
- AVFS — https://avf.sourceforge.net/
- dmesg — https://manpages.ubuntu.com/manpages/bionic/man1/dmesg.1.html
- jq — https://jqlang.github.io/jq/
- libarchive — https://libarchive.org/
- CLI11 — https://github.com/CLIUtils/CLI11

Полное дерево парсеров UniLogs



Полное UML для парсера



Пример вывода

```
kuarni@NB-6108:~/unilogs/cmake-build-debug-wsl/src/cli$ ./unilogs source -f file_with_logs.txt parse | jq
[
  {
    "log line": "kern  :info  : [Sat Mar  9 13:38:21 2024] sd 1:0:102:0: [sdkg] tag#5 FAILED Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE",
    "next": {
      "next": {
        "name": "dmesg",
        "facility": "Kern",
        "level": "Info",
        "timestamp": "Sat Mar 09 13:38:21 2024",
        "next": {
          "name": "SCSI",
          "address": {
            "host": 1,
            "bus": 0,
            "target": 102,
            "lun": 0
          },
          "device name": "sdkg",
          "next": {
            "tag": 5,
            "next": {
              "name": "Result",
              "mlret name": "FAILED",
              "mlret": "0x2003",
              "hostbyte name": "DID_OK",
              "hostbyte": "0x00",
              "driverbyte name": "DRIVER_SENSE",
              "driverbyte": "0x08",
              "cmd age": null
            }
          }
```