gitstat
Анализ и статистика по истории git

Лапшин Дмитрий
Руководитель: Кринкин Кирилл Владимирович

СПб НИАУ РАН

Весна 2015

Содержание

Цель проекта

Реализовать анализатор, находящий в проекте проблемные места, исходя из истории проекта, отслеживаемой системой контроля версий `git`.

Задачи проекта

1. Изучить систему контроля версий `git`, как она составляет изменения (`git diff`).
2. Сбор истории файлов и строк из `git` (`git show`, `git log`, `git blame`, ...).
3. Метод поиска проблемных строк в проекте на выданном диапазоне истории.
4. Реализация анализатора файла, папки.
5. Визуализация.

Задачи проекта

1. Изучить систему контроля версий `git`, как она составляет изменения (`git diff`).
2. Сбор истории файлов и строк из `git` (`git show`, `git log`, `git blame`, ...).
3. Метод поиска проблемных строк в проекте на выданном диапазоне истории.
4. Реализация анализатора файла, папки.
5. Визуализация.

Задачи проекта

1. Изучить систему контроля версий `git`, как она составляет изменения (`git diff`).
2. Сбор истории файлов и строк из `git` (`git show`, `git log`, `git blame`, ...).
3. Метод поиска проблемных строк в проекте на выданном диапазоне истории.
4. Реализация анализатора файла, папки.
5. Визуализация.

Задачи проекта

1. Изучить систему контроля версий `git`, как она составляет изменения (`git diff`).
2. Сбор истории файлов и строк из `git` (`git show`, `git log`, `git blame`, ...).
3. Метод поиска проблемных строк в проекте на выданном диапазоне истории.
4. Реализация анализатора файла, папки.
5. Визуализация.

Задачи проекта

1. Изучить систему контроля версий `git`, как она составляет изменения (`git diff`).
2. Сбор истории файлов и строк из `git` (`git show`, `git log`, `git blame`, ...).
3. Метод поиска проблемных строк в проекте на выданном диапазоне истории.
4. Реализация анализатора файла, папки.
5. Визуализация.

Проблемные строки

```
// 1 revision:
double const g = 9.8;

// this day later:
double const g = 9.80665;

// the day after:
double const g = 10;

// next week:
double const g = 2;
```

- *Не зная*, что означает `double const g`, *видно*, что что-то не так.
- Это не видно, если смотреть только на *текущую* версию кода.

Проблемные строки

```
// 1 revision:
double const g = 9.8;

// this day later:
double const g = 9.80665;

// the day after:
double const g = 10;

// next week:
double const g = 2;
```

- *Не зная*, что означает `double const g`, *видно*, что что-то не так.
- Это не видно, если смотреть только на *текущую* версию кода.

Проблемные строки

```
// 1 revision:
double const g = 9.8;

// this day later:
double const g = 9.80665;

// the day after:
double const g = 10;

// next week:
double const g = 2;
```

- *Не зная*, что означает `double const g`, *видно*, что что-то не так.
- Это не видно, если смотреть только на *текущую* версию кода.

Проблемные строки

- Проблемной имеет смысл назвать строку, которая за малое количество раз много менялась.
- Если строку целиком заменили, удалили или добавили — это не интересно.
- Строки сдвигаются по нумерации, но это их не меняет.

В качестве текущей характеристики «проблемности» строки — сколько суммарно изменений внутри строки произошло.

Проблемные строки

- Проблемной имеет смысл назвать строку, которая за малое количество раз много менялась.
- Если строку целиком заменили, удалили или добавили — это не интересно.
- Строки сдвигаются по нумерации, но это их не меняет.

В качестве текущей характеристики «проблемности» строки — сколько суммарно изменений внутри строки произошло.

Проблемные строки

- Проблемной имеет смысл назвать строку, которая за малое количество раз много менялась.
- Если строку целиком заменили, удалили или добавили — это не интересно.
- Строки сдвигаются по нумерации, но это их не меняет.

В качестве текущей характеристики «проблемности» строки — сколько суммарно изменений внутри строки произошло.

Проблемные строки

- Проблемной имеет смысл назвать строку, которая за малое количество раз много менялась.
- Если строку целиком заменили, удалили или добавили — это не интересно.
- Строки сдвигаются по нумерации, но это их не меняет.

В качестве текущей характеристики «проблемности» строки — сколько суммарно изменений внутри строки произошло.

git

В начале догло исследловался сам git, чтобы понять, что можно использовать прямо из git.

- В качестве основной команды вывода истории используется стандартная `git log` с кучей полезных флагов.
- Параллельно с ней использовался другой вызов `git log --word-diff`, который выделяет извенения именно внутри строк.
- Отслеживать историю конкретной строки git не умеет.
- Для обхода этого анализатор разбирает вывод `git log` и проходтися с помощью структуры rope (использовалось декартого дерево, treap).

git

В начале догло исследловался сам git, чтобы понять, что можно использовать прямо из git.

- В качестве основной команды вывода истории используется стандартная `git log` с кучей полезных флагов.
- Параллельно с ней использовался другой вызов `git log --word-diff`, который выделяет извенения именно внутри строк.
- Отслеживать историю конкретной строки git не умеет.
- Для обхода этого анализатор разбирает вывод `git log` и проходтися с помощью структуры rope (использовалось декартого дерево, treap).

git

В начале догло исследловался сам `git`, чтобы понять, что можно использовать прямо из `git`.

- В качестве основной команды вывода истории используется стандартная `git log` с кучей полезных флагов.
- Параллельно с ней использовался другой вызов `git log --word-diff`, который выделяет извенения именно внутри строк.
- Отслеживать историю конкретной строки `git` не умеет.
- Для обхода этого анализатор разбирает вывод `git log` и проходтися с помощью структуры `rope` (использовалось декартого дерево, `treap`).

git

В начале догло исследловался сам `git`, чтобы понять, что можно использовать прямо из `git`.

- В качестве основной команды вывода истории используется стандартная `git log` с кучей полезных флагов.
- Параллельно с ней использовался другой вызов `git log --word-diff`, который выделяет извенения именно внутри строк.
- Отслеживать историю конкретной строки `git` не умеет.
- Для обхода этого анализатор разбирает вывод `git log` и проходтися с помощью структуры `rope` (использовалось декартого дерево, `treap`).

git

В начале догло исследловался сам git, чтобы понять, что можно использовать прямо из git.

- В качестве основной команды вывода истории используется стандартная `git log` с кучей полезных флагов.
- Параллельно с ней использовался другой вызов `git log --word-diff`, который выделяет извенения именно внутри строк.
- Отслеживать историю конкретной строки git не умеет.
- Для обхода этого анализатор разбирает вывод `git log` и проходтися с помощью структуры `rope` (использовалось декартого дерево, `treap`).

Объединение и разделение строк

`word-diff` иногда начинает объединять или разделять строки:

```
r = ioctl(hdev->control, VHOST_SET_LOG_BASE, // Enter
          (uint64_t)(unsigned long)hdev->log);

r = hdev->vhost_ops->vhost_call(hdev, VHOST_SET_LOG_BASE,
hdev->log); // Одна строка
```

И если с таким случаем разобраться получилось...

Объединение и разделение строк

word-diff иногда начинает объединять или разделять строки:

```
r = ioctl(hdev->control, VHOST_SET_LOG_BASE, // Enter
          (uint64_t)(unsigned long)hdev->log);
```

```
r = hdev->vhost_ops->vhost_call(hdev, VHOST_SET_LOG_BASE,
hdev->log); // Одна строка
```

И если с таким случаем разобраться получилось...

Объединение и разделение строк

word-diff иногда начинает объединять или разделять строки:

```
r = ioctl(hdev->control, VHOST_SET_LOG_BASE, // Enter
          (uint64_t)(unsigned long)hdev->log);

r = hdev->vhost_ops->vhost_call(hdev, VHOST_SET_LOG_BASE,
hdev->log); // Одна строка
```

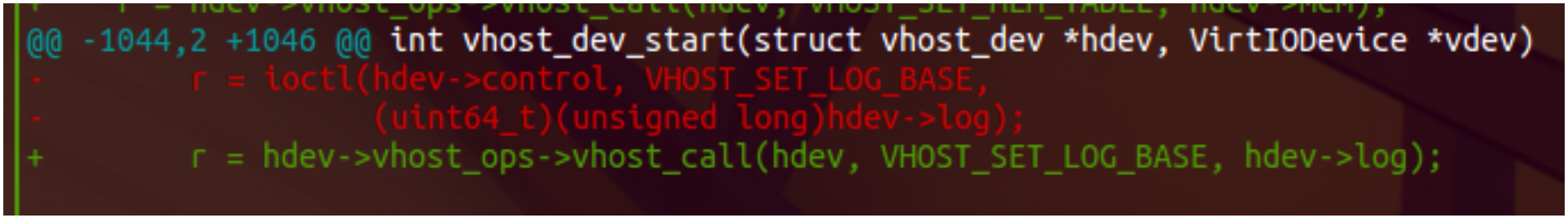
И если с таким случаем разобраться получилось...

Объединение и разделение строк

word-diff иногда начинает объединять или разделять строки:

```
r = ioctl(hdev->control, VHOST_SET_LOG_BASE, // Enter
          (uint64_t)(unsigned long)hdev->log);
```

```
r = hdev->vhost_ops->vhost_call(hdev, VHOST_SET_LOG_BASE,
hdev->log); // Одна строка
```



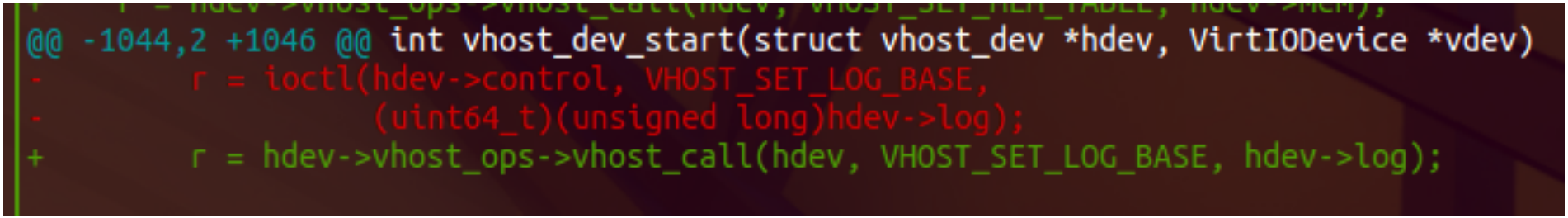
И если с таким случаем разобраться получилось...

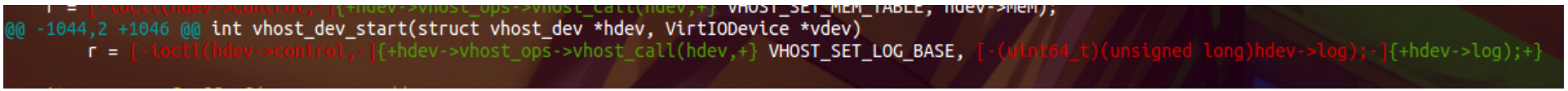
Объединение и разделение строк

word-diff иногда начинает объединять или разделять строки:

```
r = ioctl(hdev->control, VHOST_SET_LOG_BASE, // Enter
          (uint64_t)(unsigned long)hdev->log);
```

```
r = hdev->vhost_ops->vhost_call(hdev, VHOST_SET_LOG_BASE,
hdev->log); // Одна строка
```





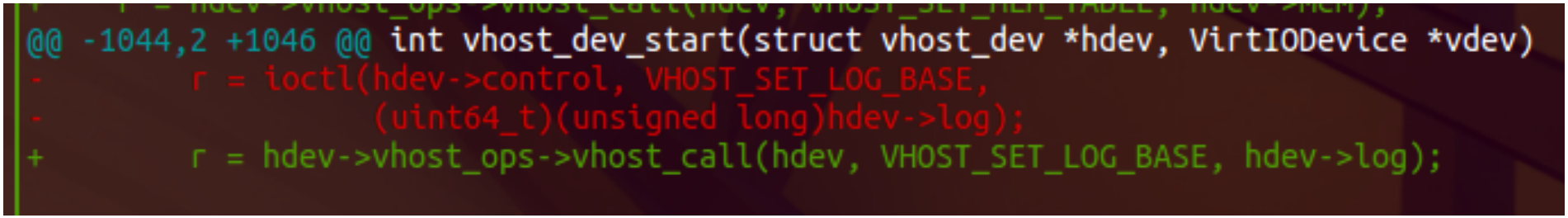
И если с таким случаем разобраться получилось...

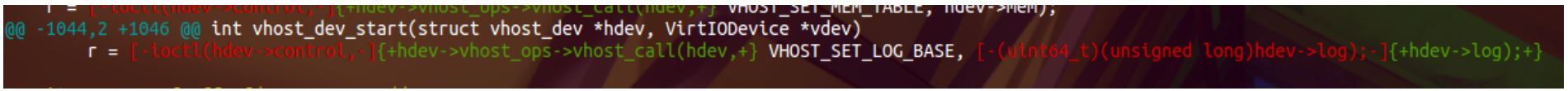
Объединение и разделение строк

word-diff иногда начинает объединять или разделять строки:

```
r = ioctl(hdev->control, VHOST_SET_LOG_BASE, // Enter
          (uint64_t)(unsigned long)hdev->log);
```

```
r = hdev->vhost_ops->vhost_call(hdev, VHOST_SET_LOG_BASE,
hdev->log); // Одна строка
```





И если с таким случаем разобраться получилось...

Объединение и разделение строк

...то с таким — нет:

```
while ((bit = sizeof(log) > sizeof(int) ?
        ffsll(log) : ffs(log))) {
```

```
while (log) {
    int bit = ctzl(log);
```

Тут `--word-diff` начинает смешивать строки в изменениях.
Как именно работает режим `--word-diff` нигде не описано.
Кроме этого, были обнаружены ещё несколько странных особенностей вывода `git log --word-diff`.

Объединение и разделение строк

...то с таким — нет:

```
while ((bit = sizeof(log) > sizeof(int) ?
        ffsll(log) : ffs(log))) {
```

```
while (log) {
    int bit = ctzl(log);
```

Тут `--word-diff` начинает смешивать строки в изменениях.
Как именно работает режим `--word-diff` нигде не описано.
Кроме этого, были обнаружены ещё несколько странных особенностей вывода `git log --word-diff`.

Объединение и разделение строк

...то с таким — нет:

```
while ((bit = sizeof(log) > sizeof(int) ?
        ffsll(log) : ffs(log))) {
```

```
while (log) {
    int bit = ctzl(log);
```

Тут `--word-diff` начинает смешивать строки в изменениях.
Как именно работает режим `--word-diff` нигде не описано.
Кроме этого, были обнаружены ещё несколько странных особенностей вывода `git log --word-diff`.

Объединение и разделение строк

...то с таким — нет:

```
while ((bit = sizeof(log) > sizeof(int) ?
        ffsll(log) : ffs(log))) {
```

```
while (log) {
    int bit = ctzl(log);
```

Тут `--word-diff` начинает смешивать строки в изменениях.

Как именно работает режим `--word-diff` нигде не описано.

Кроме этого, были обнаружены ещё несколько странных особенностей вывода `git log --word-diff`.

Объединение и разделение строк

...то с таким — нет:

```
while ((bit = sizeof(log) > sizeof(int) ?
        ffsll(log) : ffs(log))) {
```

```
while (log) {
    int bit = ctzl(log);
```

Тут `--word-diff` начинает смешивать строки в изменениях.

Как именно работает режим `--word-diff` нигде не описано.

Кроме этого, были обнаружены ещё несколько странных особенностей вывода `git log --word-diff`.

Объединение и разделение строк

...то с таким — нет:

```
while ((bit = sizeof(log) > sizeof(int) ?
        ffsll(log) : ffs(log))) {
```

```
while (log) {
    int bit = ctzl(log);
```

Тут `--word-diff` начинает смешивать строки в изменениях.
Как именно работает режим `--word-diff` нигде не описано.
Кроме этого, были обнаружены ещё несколько странных особенностей вывода `git log --word-diff`.

Что удалось

- Разобраться, как `git` работает с историей на довольно глубоком уровне.
- В целом много узнать о механизмах `git`-а (взаимотоношения коммитов, идеология патчей, виды и режимы файлов...).
- Разобрать крайне сложно устроенный вывод.
- Реализация подсчёта изменений в одном файле реализована и работает, пока не налетает на ошибку выше.

Что удалось

- Разобраться, как `git` работает с историей на довольно глубоком уровне.
- В целом много узнать о механизмах `git`-а (взаимотоношения коммитов, идеология патчей, виды и режимы файлов...).
- Разобрать крайне сложно устроенный вывод.
- Реализация подсчёта изменений в одном файле реализована и работает, пока не налетает на ошибку выше.

Что удалось

- Разобраться, как `git` работает с историей на довольно глубоком уровне.
- В целом много узнать о механизмах `git`-а (взаимотоношения коммитов, идеология патчей, виды и режимы файлов...).
- Разобрать крайне сложно устроенный вывод.
- Реализация подсчёта изменений в одном файле реализована и работает, пока не налетает на ошибку выше.

Что удалось

- Разобраться, как `git` работает с историей на довольно глубоком уровне.
- В целом много узнать о механизмах `git`-а (взаимотоношения коммитов, идеология патчей, виды и режимы файлов...).
- Разобрать крайне сложно устроенный вывод.
- Реализация подсчёта изменений в одном файле реализована и работает, пока не налетает на ошибку выше.

Что удалось

- Разобраться, как `git` работает с историей на довольно глубоком уровне.
- В целом много узнать о механизмах `git`-а (взаимотоношения коммитов, идеология патчей, виды и режимы файлов...).
- Разобрать крайне сложно устроенный вывод.
- Реализация подсчёта изменений в одном файле реализована и работает, пока не налетает на ошибку выше.

Что ещё не удалось / Планы на будущее

Большинство этих задач можно решить в ближайшем будущем:

- Анализировать сразу папки.
- Визуализатор.

Эти задачи требуют уже больших усилий:

- Разобрать анализ `git`-ом фрагментов, описанных выше.
- Возможно, стоит рассмотреть дополнительные классы изменений (переименовывание файлов, ...) и улучшить разбиение на слова.

Что ещё не удалось / Планы на будущее

Большинство этих задач можно решить в ближайшем будущем:

- Анализировать сразу папки.
- Визуализатор.

Эти задачи требуют уже больших усилий:

- Разобрать анализ `git`-ом фрагментов, описанных выше.
- Возможно, стоит рассмотреть дополнительные классы изменений (переименовывание файлов, ...) и улучшить разбиение на слова.

Что ещё не удалось / Планы на будущее

Большинство этих задач можно решить в ближайшем будущем:

- Анализировать сразу папки.
- Визуализатор.

Эти задачи требуют уже больших усилий:

- Разобрать анализ `git`-ом фрагментов, описанных выше.
- Возможно, стоит рассмотреть дополнительные классы изменений (переименовывание файлов, ...) и улучшить разбиение на слова.

Что ещё не удалось / Планы на будущее

Большинство этих задач можно решить в ближайшем будущем:

- Анализировать сразу папки.
- Визуализатор.

Эти задачи требуют уже больших усилий:

- Разобрать анализ `git`-ом фрагментов, описанных выше.
- Возможно, стоит рассмотреть дополнительные классы изменений (переименовывание файлов, ...) и улучшить разбиение на слова.

Что ещё не удалось / Планы на будущее

Большинство этих задач можно решить в ближайшем будущем:

- Анализировать сразу папки.
- Визуализатор.

Эти задачи требуют уже больших усилий:

- Разобрать анализ `git`-ом фрагментов, описанных выше.
- Возможно, стоит рассмотреть дополнительные классы изменений (переименовывание файлов, ...) и улучшить разбиение на слова.

Выводы

- *git — stupid container tracker*
  git действительно мало понимает в отслеживаемых файлах. Для лучшего анализа его необходимо настраивать на каждый формат файла, который он хранит.
- Реализация git практически не документирована.
- Тем не менее, git — очень мощная система с множеством опций и методов модификации.
- Задача анализа истории по git действительно сложна, но решаема.
- Python — великолепный язык для взаимодействия с командами и их выводами.

Выводы

- *git — stupid container tracker*
  `git` действительно мало понимает в отслеживаемых файлах. Для лучшего анализа его необходимо настраивать на каждый формат файла, который он хранит.
- Реализация `git` практически не документирована.
- Тем не менее, `git` — очень мощная система с множеством опций и методов модификации.
- Задача анализа истории по `git` действительно сложна, но решаема.
- Python — великолепный язык для взаимодействия с командами и их выводами.

Выводы

- *git — stupid container tracker*
  `git` действительно мало понимает в отслеживаемых файлах. Для лучшего анализа его необходимо настраивать на каждый формат файла, который он хранит.
- Реализация `git` практически не документирована.
- Тем не менее, `git` — очень мощная система с множеством опций и методов модификации.
- Задача анализа истории по `git` действительно сложна, но решаема.
- Python — великолепный язык для взаимодействия с командами и их выводами.

Выводы

- *git — stupid container tracker*
  `git` действительно мало понимает в отслеживаемых файлах. Для лучшего анализа его необходимо настраивать на каждый формат файла, который он хранит.
- Реализация `git` практически не документирована.
- Тем не менее, `git` — очень мощная система с множеством опций и методов модификации.
- Задача анализа истории по `git` действительно сложна, но решаема.
- Python — великолепный язык для взаимодействия с командами и их выводами.

Выводы

- *git — stupid container tracker*
  `git` действительно мало понимает в отслеживаемых файлах. Для лучшего анализа его необходимо настраивать на каждый формат файла, который он хранит.
- Реализация `git` практически не документирована.
- Тем не менее, `git` — очень мощная система с множеством опций и методов модификации.
- Задача анализа истории по `git` действительно сложна, но решаема.
- Python — великолепный язык для взаимодействия с командами и их выводами.

Выводы

- *git — stupid container tracker*
  `git` действительно мало понимает в отслеживаемых файлах. Для лучшего анализа его необходимо настраивать на каждый формат файла, который он хранит.
- Реализация `git` практически не документирована.
- Тем не менее, `git` — очень мощная система с множеством опций и методов модификации.
- Задача анализа истории по `git` действительно сложна, но решаема.
- Python — великолепный язык для взаимодействия с командами и их выводами.

Чему я научлся

- Глубокая работа с `git`.
- Чтение большого объёма непонятной документации, которую никто не читает.
- Регулярные выражения
- Python

Чему я научлся

- Глубокая работа с `git`.
- Чтение большого объёма непонятной документации, которую никто не читает.
- Регулярные выражения
- Python

Чему я научлся

- Глубокая работа с `git`.
- Чтение большого объёма непонятной документации, которую никто не читает.
- Регулярные выражения
- Python

Чему я научлся

- Глубокая работа с `git`.
- Чтение большого объёма непонятной документации, которую никто не читает.
- Регулярные выражения
- Python

Чему я научлся

- Глубокая работа с `git`.
- Чтение большого объёма непонятной документации, которую никто не читает.
- Регулярные выражения
- Python

Спасибо за внимание.

Репозиторий: `https://github.com/OSLL/edu-git-stats`

В качестве примеров были взяты строки из
`https://github.com/OSLL/qemu-xtensa/`