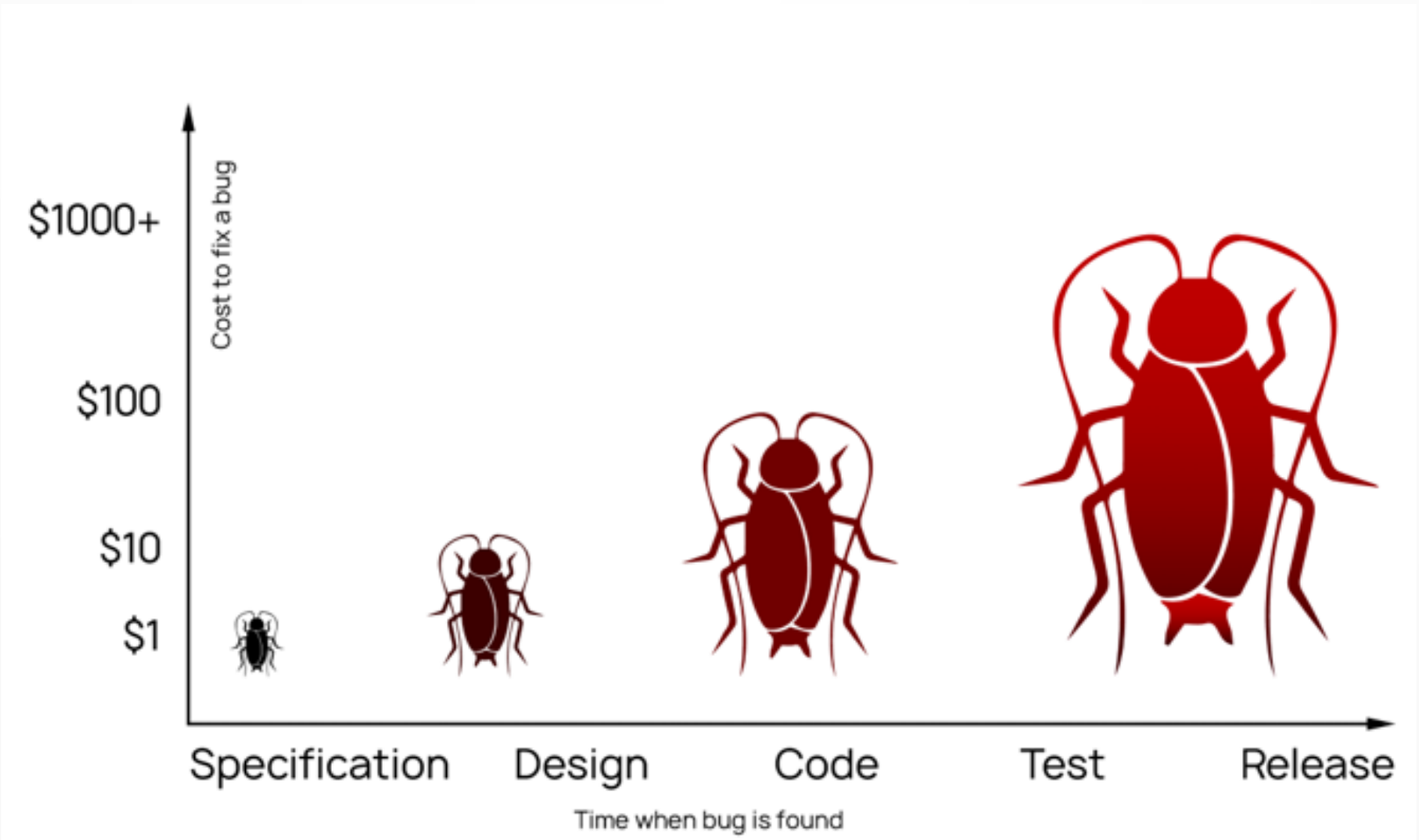
Улучшение качества открытого программного обеспечения с помощью инструментов анализа кода

Докладчик: Звягинцев Максим

<max.post.space@gmail.com>

Наши насущные проблемы

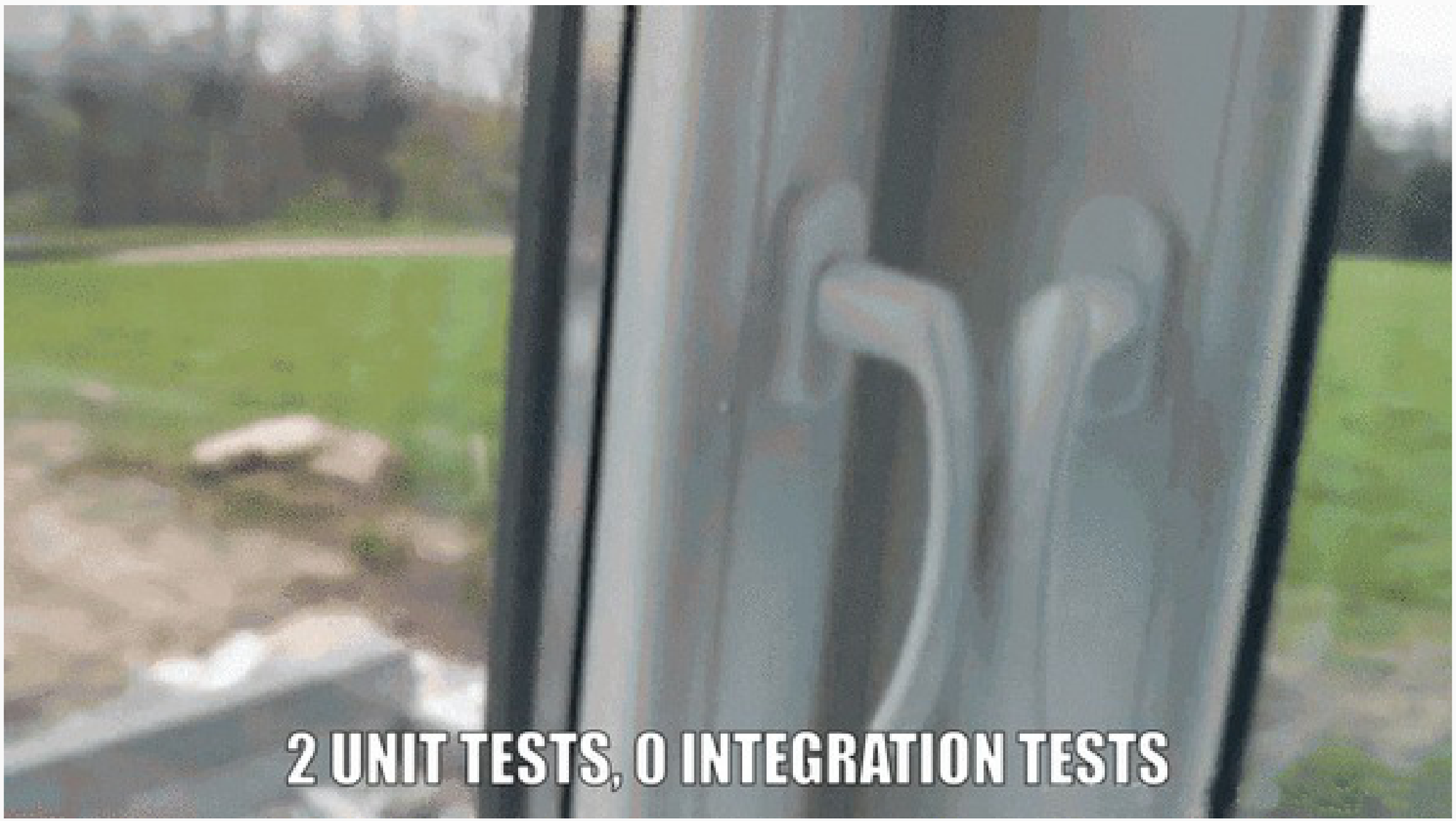
Стоимость исправления ошибок



Ревью кода

Модульное тестирование

Интеграционное тестирование



Динамические анализаторы

- Valgrind
- Sanitizers (Address Sanitizer, Thread Sanitizer, etc.)
- Intel Thread Checker

Valgrind

- Memcheck
- Massif
- Helgrind
- Callgrind

Valgrind

- Memcheck
- Massif
- Helgrind
- Callgrind

Memcheck (тестовый код)

```
#include <iostream>
int* get_array(size_t size)
{
  int *array = new int[size];
  for (int i = 0; i < size; i++)
    array[i] = i;
  return array;
}

int main()
{
  const size_t size = 5;
  int *array = get_array(size);

  for (int i = 0; i < size; i++)
    std::cout << array[i] << std::endl;

  return 0;
}
```

Memcheck (вывод)

```
==83== Memcheck, a memory error detector
==83== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==83== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==83== Command: ./test
==83==
==83== error calling PR_SET_PTRACER, vgdb might block
0
1
2
3
4
==83==
==83== HEAP SUMMARY:
==83==     in use at exit: 20 bytes in 1 blocks
==83==   total heap usage: 3 allocs, 2 frees, 73,748 bytes allocated
==83==
==83== LEAK SUMMARY:
==83==    definitely lost: 20 bytes in 1 blocks
==83==    indirectly lost: 0 bytes in 0 blocks
==83==      possibly lost: 0 bytes in 0 blocks
==83==    still reachable: 0 bytes in 0 blocks
==83==         suppressed: 0 bytes in 0 blocks
==83== Rerun with --leak-check=full to see details of leaked memory
==83==
==83== For counts of detected and suppressed errors, rerun with: -v
==83== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Memcheck (вывод)

```
==83==
==83== HEAP SUMMARY:
==83==     in use at exit: 20 bytes in 1 blocks
==83==   total heap usage: 3 allocs, 2 frees, 73,748 bytes allocated
==83==
==83== LEAK SUMMARY:
==83==    definitely lost: 20 bytes in 1 blocks
==83==    indirectly lost: 0 bytes in 0 blocks
==83==      possibly lost: 0 bytes in 0 blocks
==83==    still reachable: 0 bytes in 0 blocks
==83==         suppressed: 0 bytes in 0 blocks
==83== Rerun with --leak-check=full to see details of leaked memory
```

Memcheck (тестовый код)

```
#include <iostream>

int* get_array(size_t size)
{
  int *array = new int[size];
  for (int i = 0; i < size; i++)
    array[i] = i;
  return array;
}

int main()
{
  const size_t size = 5;
  int *array = get_array(size);

  for (int i = 0; i < size; i++)
    std::cout << array[i] << std::endl;

  delete[] array;
  return 0;
}
```

Memcheck (исправленный вывод)

```
==372== HEAP SUMMARY:
==372==     in use at exit: 0 bytes in 0 blocks
==372==   total heap usage: 3 allocs, 3 frees, 73,748 bytes allocated
==372==
==372== All heap blocks were freed -- no leaks are possible
==372==
==372== For counts of detected and suppressed errors, rerun with: -v
==372== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Callgrind (тестовый код)

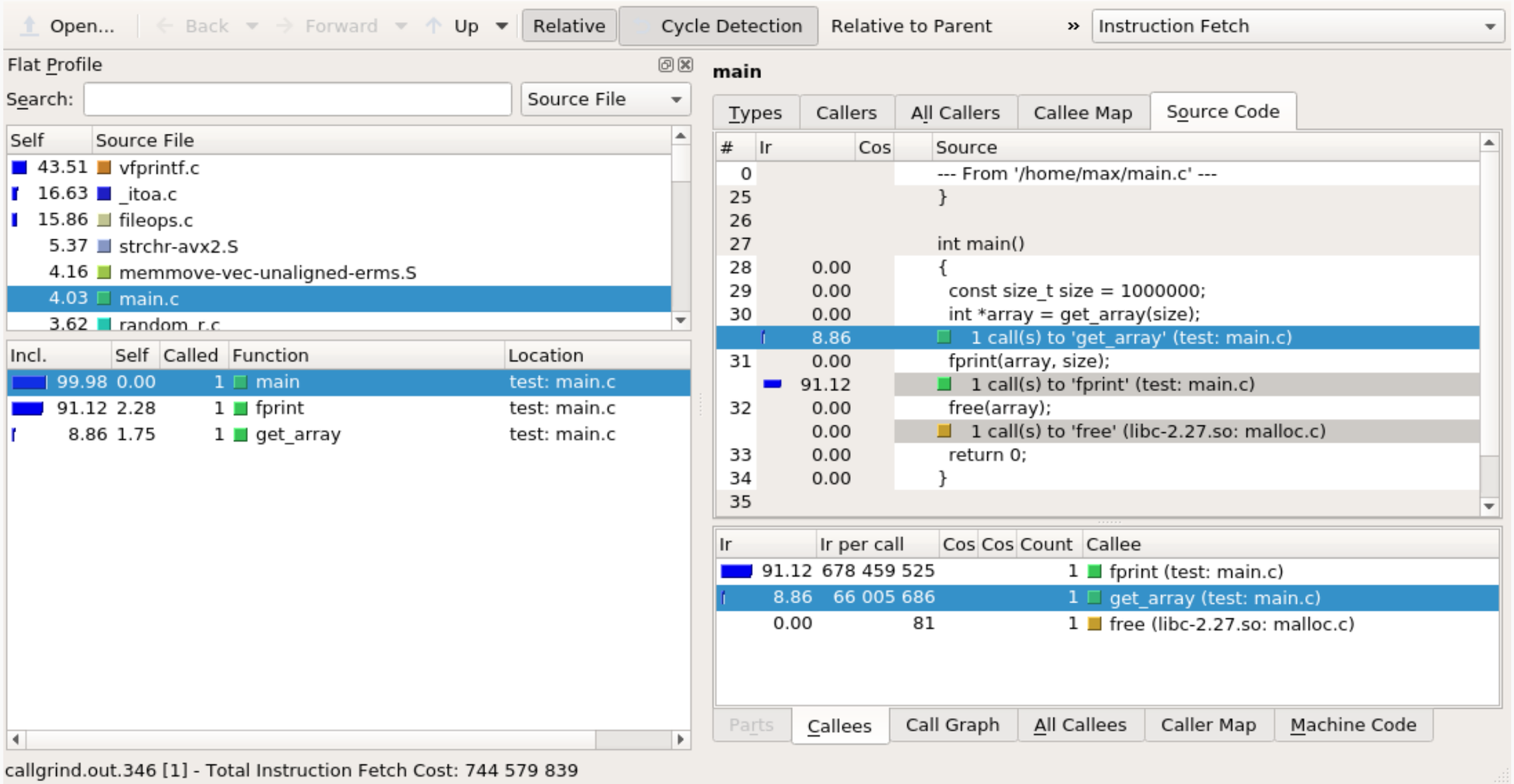
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int* get_array(size_t size)
{
  int *array = malloc(size*sizeof(int));

  srand (time(NULL));
  for (int i = 0; i < size; i++)
    array[i] = rand();

  return array;
}
```

```c
void fprint (int *array, size_t size)
{
  FILE *output;
  output = fopen ("output", "wb");

  for (int i = 0; i < size; i++)
    fprintf(output, "%i\n", array[i]);

  fclose (output);
}

int main()
{
  const size_t size = 1000000;
  int *array = get_array(size);
  fprint(array, size);
  free(array);
  return 0;
}
```

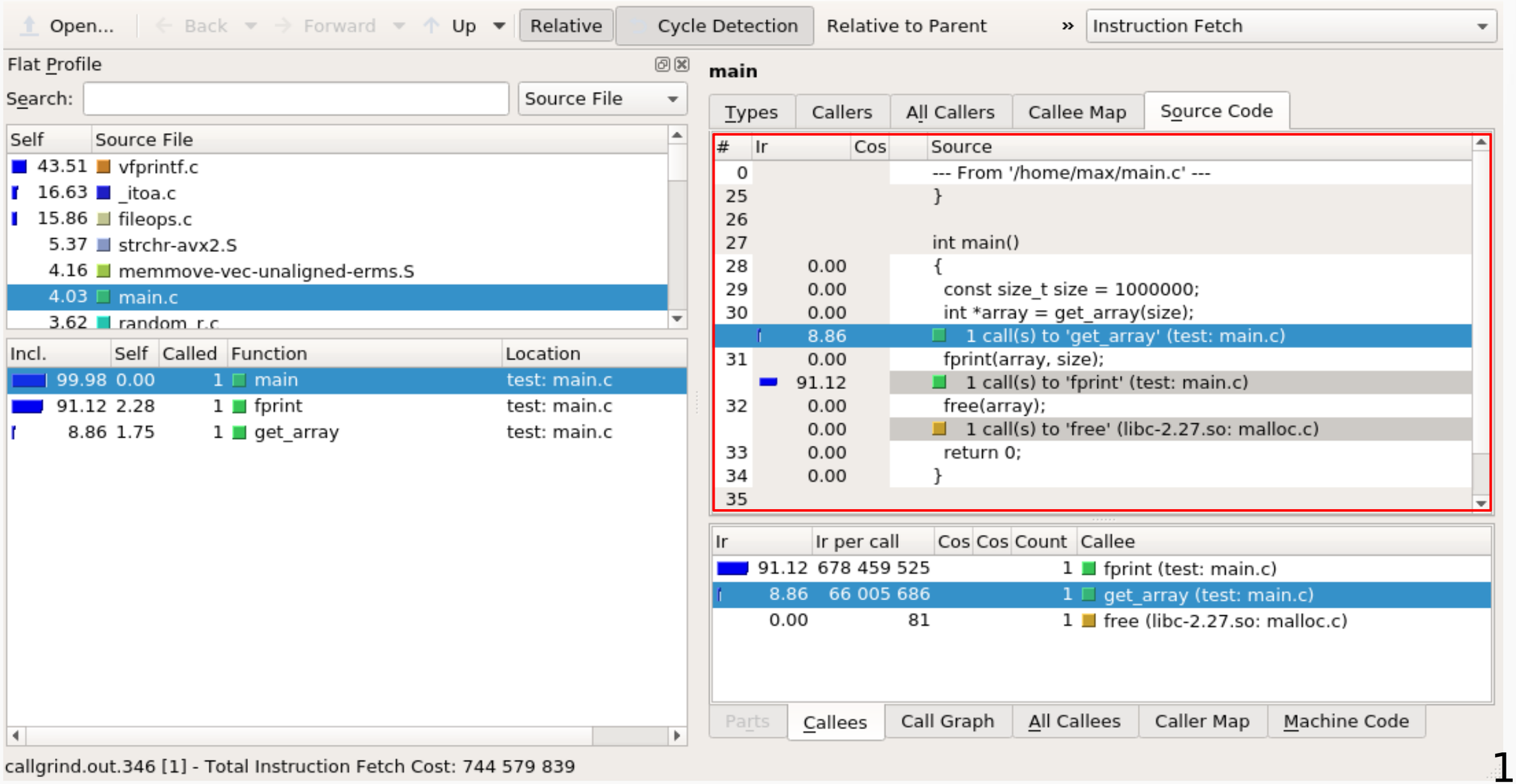
Callgrind (вывод)

```
max@ZVYAGINTSEV-PC:~$ valgrind --tool=callgrind --simulate-cache=yes ./test
==346== Callgrind, a call-graph generating cache profiler
==346== Copyright (C) 2002-2017, and GNU GPL'd, by Josef Weidendorfer et al.
==346== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==346== Command: ./test
==346==
--346-- warning: L3 cache found, using its data for the LL simulation.
==346== For interactive control, run 'callgrind_control -h'.
==346== error calling PR_SET_PTRACER, vgdb might block
==346==
==346== Events    : Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw
==346== Collected : 744579839 169554513 118193786 1165 65047 63177 1145 2036 63084
==346==
==346== I   refs:      744,579,839
==346== I1  misses:          1,165
==346== LLi misses:          1,145
==346== I1  miss rate:        0.00%
==346== LLi miss rate:        0.00%
==346==
==346== D   refs:      287,748,299  (169,554,513 rd + 118,193,786 wr)
==346== D1  misses:        128,224  (     65,047 rd +      63,177 wr)
==346== LLd misses:         65,120  (      2,036 rd +      63,084 wr)
==346== D1  miss rate:         0.0% (        0.0%   +         0.1%  )
==346== LLd miss rate:         0.0% (        0.0%   +         0.1%  )
==346==
==346== LL refs:           129,389  (     66,212 rd +      63,177 wr)
==346== LL misses:          66,265  (      3,181 rd +      63,084 wr)
==346== LL miss rate:          0.0% (        0.0%   +         0.1%  )
```

KCachegrind



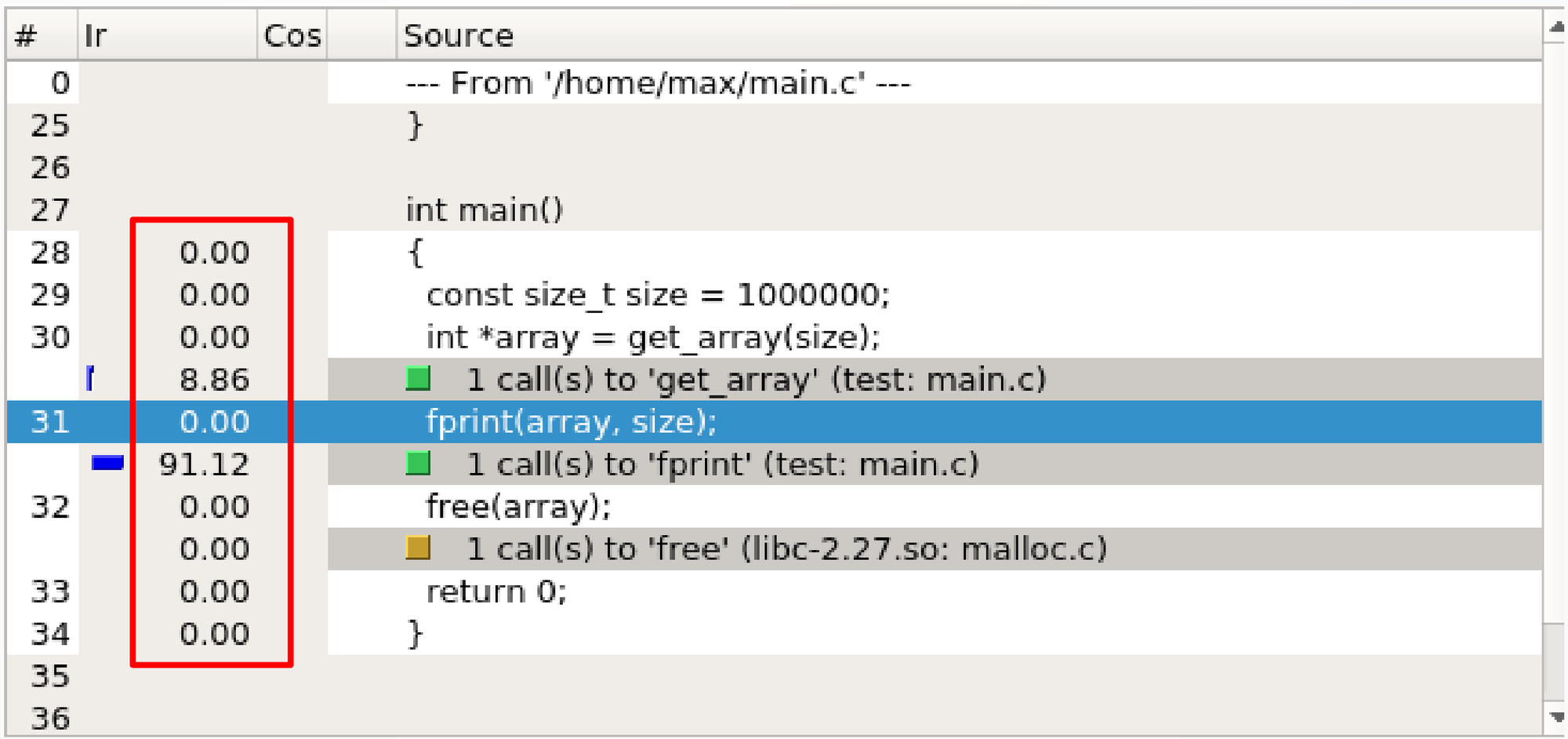
KCachegrind



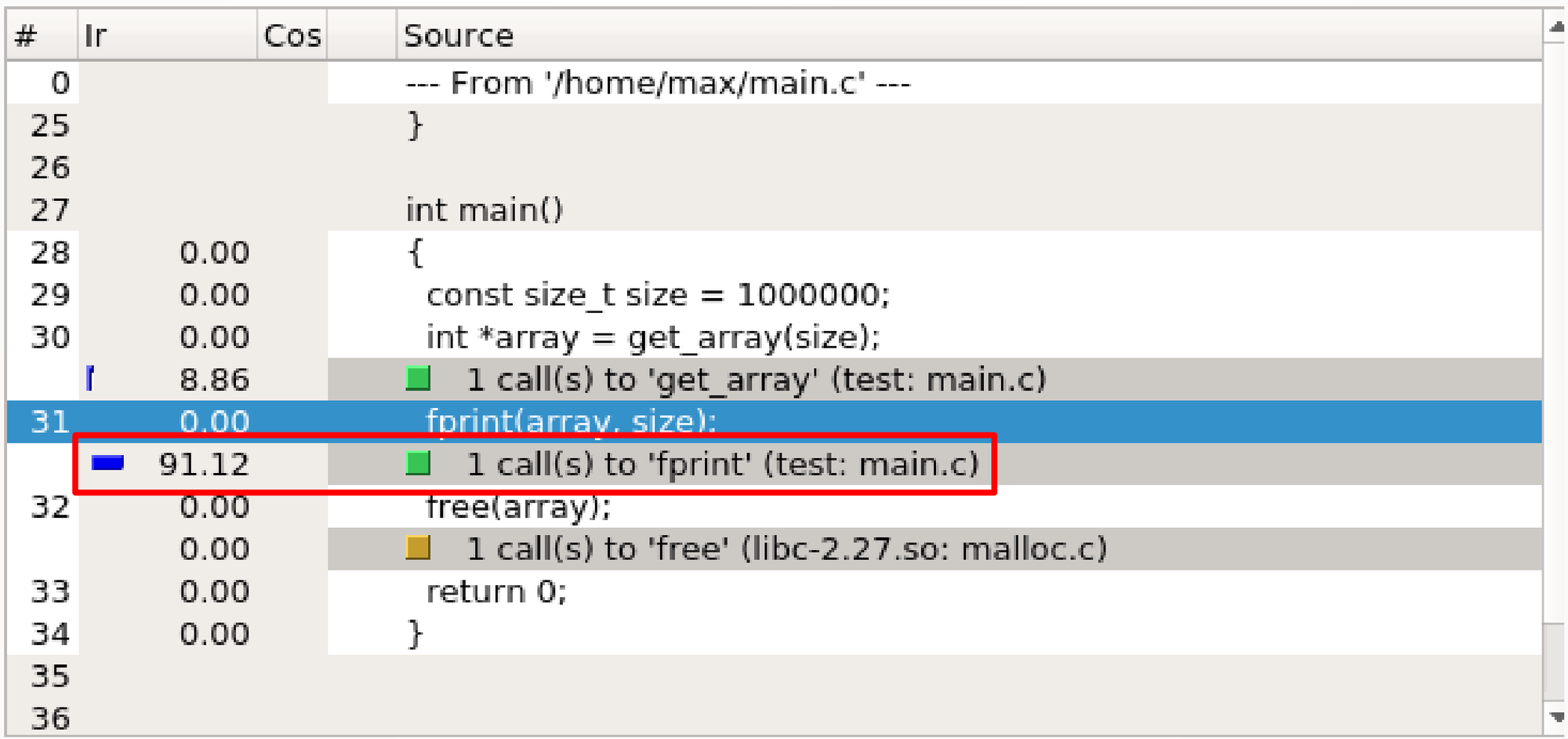
KCachegrind

#	Ir	Cos		Source
0				--- From '/home/max/main.c' ---
25				}
26				
27				int main()
28		0.00		{
29		0.00		const size_t size = 1000000;
30		0.00		int *array = get_array(size);
		8.86		1 call(s) to 'get_array' (test: main.c)
31		0.00		fprint(array, size);
		91.12		1 call(s) to 'fprint' (test: main.c)
32		0.00		free(array);
		0.00		1 call(s) to 'free' (libc-2.27.so: malloc.c)
33		0.00		return 0;
34		0.00		}
35				
36				

KCachegrind



KCachegrind



Некоторые методы выявления ошибок

- Ревью кода
- Модульное тестирование
- Интеграционное тестирование
- Динамический анализ

Некоторые методы выявления ошибок

- Ревью кода
- Модульное тестирование
- Интеграционное тестирование
- Динамический анализ
- Статический анализ

Статические анализаторы

- Lint (бесплатно)
- Cppcheck (бесплатно)
- Clang Static Analyzer (бесплатно)
- BLAST (бесплатно)
- Roslynator (бесплатно)
- PVS-Studio (бесплатно для открытых проектов)
- Coverity (бесплатно для открытых проектов)

Ошибки на примере CMake

Примеры работы статического анализатора

```
void cmMakefile::RemoveVariablesInString(std::string& source,
                                         bool atOnly) const
{
  if (!atOnly) {
    cmsys::RegularExpression var("(\\${[A-Za-z_0-9]*})");
    while (var.find(source)) {
      source.erase(var.start(), var.end() - var.start());
    }
  }

  if (!atOnly) {
    cmsys::RegularExpression varb("(\\$ENV{[A-Za-z_0-9]*})");
    while (varb.find(source)) {
      source.erase(varb.start(), varb.end() - varb.start());
    }
  }
  cmsys::RegularExpression var2("(@[A-Za-z_0-9]*@)");
  while (var2.find(source)) {
    source.erase(var2.start(), var2.end() - var2.start());
  }
}
```

V581 The conditional expressions of the 'if' statements situated alongside each other are identical. Check lines: 3230, 3237.

Примеры работы статического анализатора

```
const char* defaultCStandard = this→GetDefinition("CMAKE_C_STANDARD_DEFAULT");
if (!defaultCStandard)
{
  . . . .
  return true;
}

. . . .

const char* existingCStandard = target->GetProperty("C_STANDARD");
if (!existingCStandard)
{
  existingCStandard = defaultCStandard;
}
. . . .

const char* const* existingCIt = existingCStandard ? std::find_if(. . . .)
                                                   : cm::cend(C_STANDARDS);
```

V547 Expression 'existingCStandard' is always true.

Примеры работы статического анализатора

```
const char* defaultCStandard = this→GetDefinition("CMAKE_C_STANDARD_DEFAULT");
if (!defaultCStandard)
{
  . . . .
  return true;
}

. . . .

const char* existingCStandard = target->GetProperty("C_STANDARD");
if (!existingCStandard)
{
  existingCStandard = defaultCStandard;
}
. . . .

const char* const* existingCIt = existingCStandard ? std::find_if(. . . .)
                                                   : cm::cend(C_STANDARDS);
```

V547 Expression 'existingCStandard' is always true.

Примеры работы статического анализатора

```
const char* defaultCStandard = this→GetDefinition("CMAKE_C_STANDARD_DEFAULT");
if (!defaultCStandard)
{
  . . . .
  return true;
}

. . . .

const char* existingCStandard = target->GetProperty("C_STANDARD");
if (!existingCStandard)
{
  existingCStandard = defaultCStandard;
}
. . . .

const char* const* existingCIt = existingCStandard ? std::find_if(. . . .)
                                                   : cm::cend(C_STANDARDS);
```

V547 Expression 'existingCStandard' is always true.

Примеры работы статического анализатора

```
if (step == 0) {
  if (start > stop) {
    step = -1;
  } else {
    step = 1;
  }
}

if ((start > stop && step > 0) || (start < stop && step < 0) ||
     step == 0) {
  . . . .
}
```

Примеры работы статического анализатора

```
if (step == 0) {
  if (start > stop) {
    step = -1;
  } else {
    step = 1;
  }
}

if ((start > stop && step > 0) || (start < stop && step < 0) ||
     step == 0) {
  . . . .
}
```

V560 A part of conditional expression is always false: step == 0.

Совместная работа

Динамический анализ

- Анализ используемых ресурсов
- Степень покрытия кода тестами
- Различные ошибки работы с памятью
- Различные ошибки, связанные с многопоточным программированием

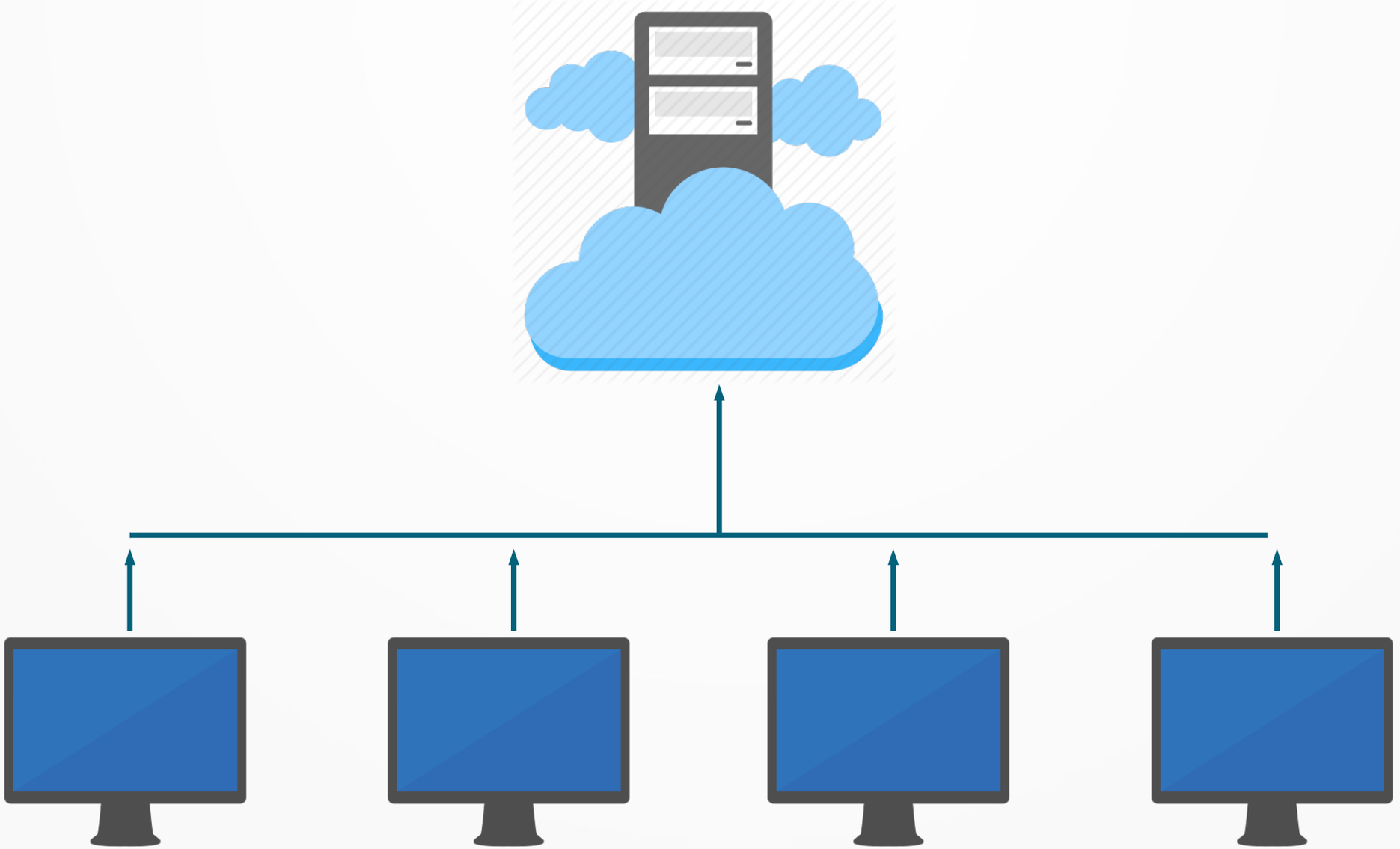
+

Статический анализ

- Ошибки копипасты
- Некорректный вызов библиотечной функции
- Проблемы кроссплатформенности
- Ошибки в повторяющемся коде
- Недостижимый, мертвый код

=

Что дальше?



CI

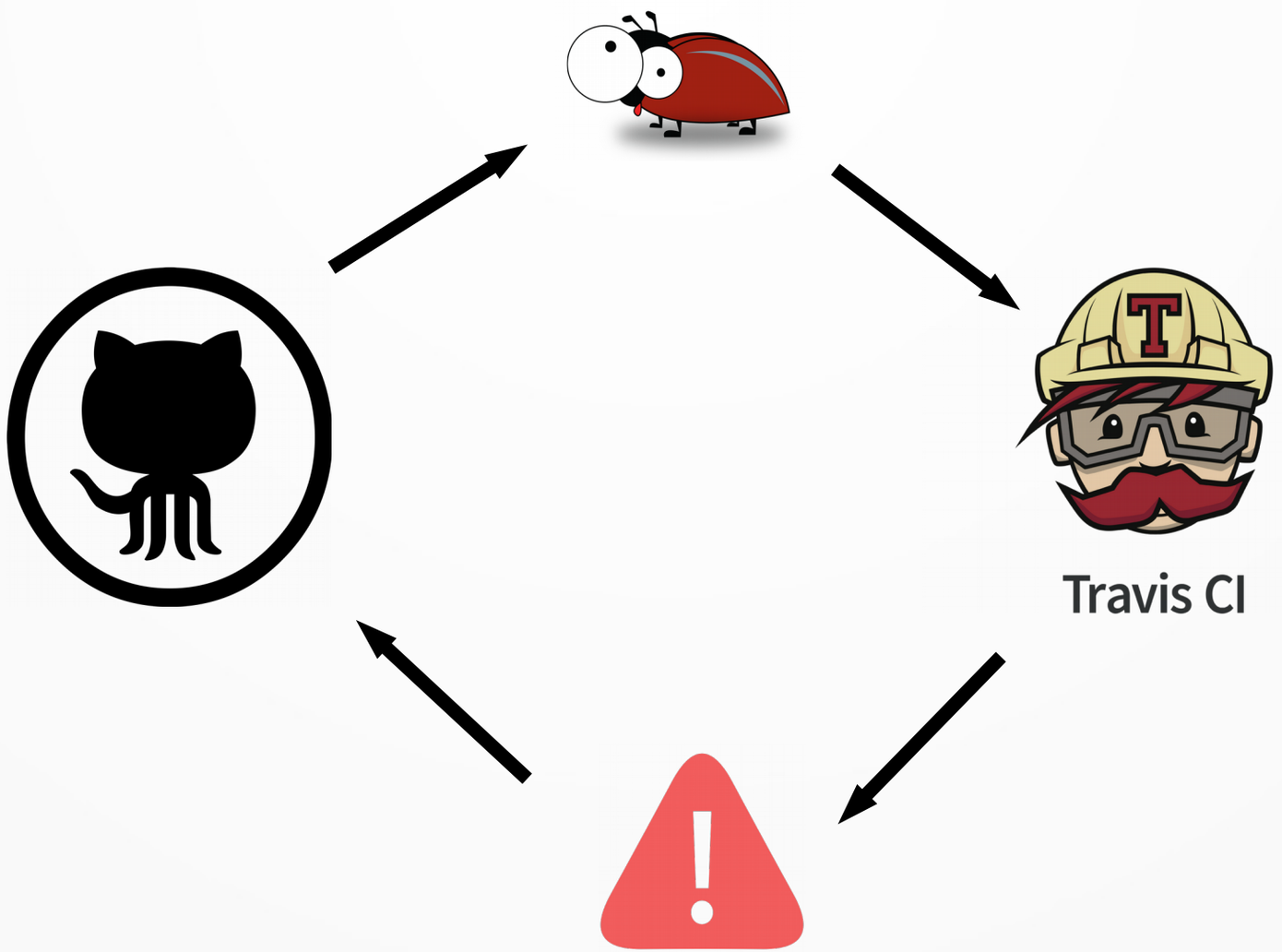
Облачные решения:

- Travis CI
- Azure DevOps
- Buddy
- Circle CI
- AppVeyor

Self-Hosted:

- Jenkins
- GitLab
- TeamCity

Travis CI



Travis CI Настройка

```
language: cpp
dist: xenial

addons:
  apt:
    update: true
    packages:
      - build-essential
      - cmake
      - googletest
      - ctest
      - jq
      - wget

cache:
    directories:
      - .PVS-Studio/

matrix:
  include:
    - os: linux
      compiler: "gcc"
      env: PVS_ANALYZE=Yes
```

Travis CI Настройка

```
language: cpp
dist: xenial

addons:
  apt:
    update: true
    packages:
      - build-essential
      - cmake
      - googletest
      - ctest
      - jq
      - wget

cache:
    directories:
      - .PVS-Studio/

matrix:
  include:
    - os: linux
      compiler: "gcc"
      env: PVS_ANALYZE=Yes
```

Travis CI Настройка

```
language: cpp
dist: xenial

addons:
  apt:
    update: true
    packages:
      - build-essential
      - cmake
      - googletest
      - ctest
      - jq
      - wget

cache:
    directories:
      - .PVS-Studio/

matrix:
  include:
    - os: linux
      compiler: "gcc"
      env: PVS_ANALYZE=Yes
```

Travis CI Настройка

```
language: cpp
dist: xenial

addons:
  apt:
    update: true
    packages:
      - build-essential
      - cmake
      - googletest
      - ctest
      - jq
      - wget

cache:
    directories:
      - .PVS-Studio/

matrix:
  include:
    - os: linux
      compiler: "gcc"
      env: PVS_ANALYZE=Yes
```

Travis CI Настройка

```
language: cpp
dist: xenial

addons:
  apt:
    update: true
    packages:
      - build-essential
      - cmake
      - googletest
      - ctest
      - jq
      - wget

cache:
    directories:
      - .PVS-Studio/

matrix:
  include:
    - os: linux
      compiler: "gcc"
      env: PVS_ANALYZE=Yes
```

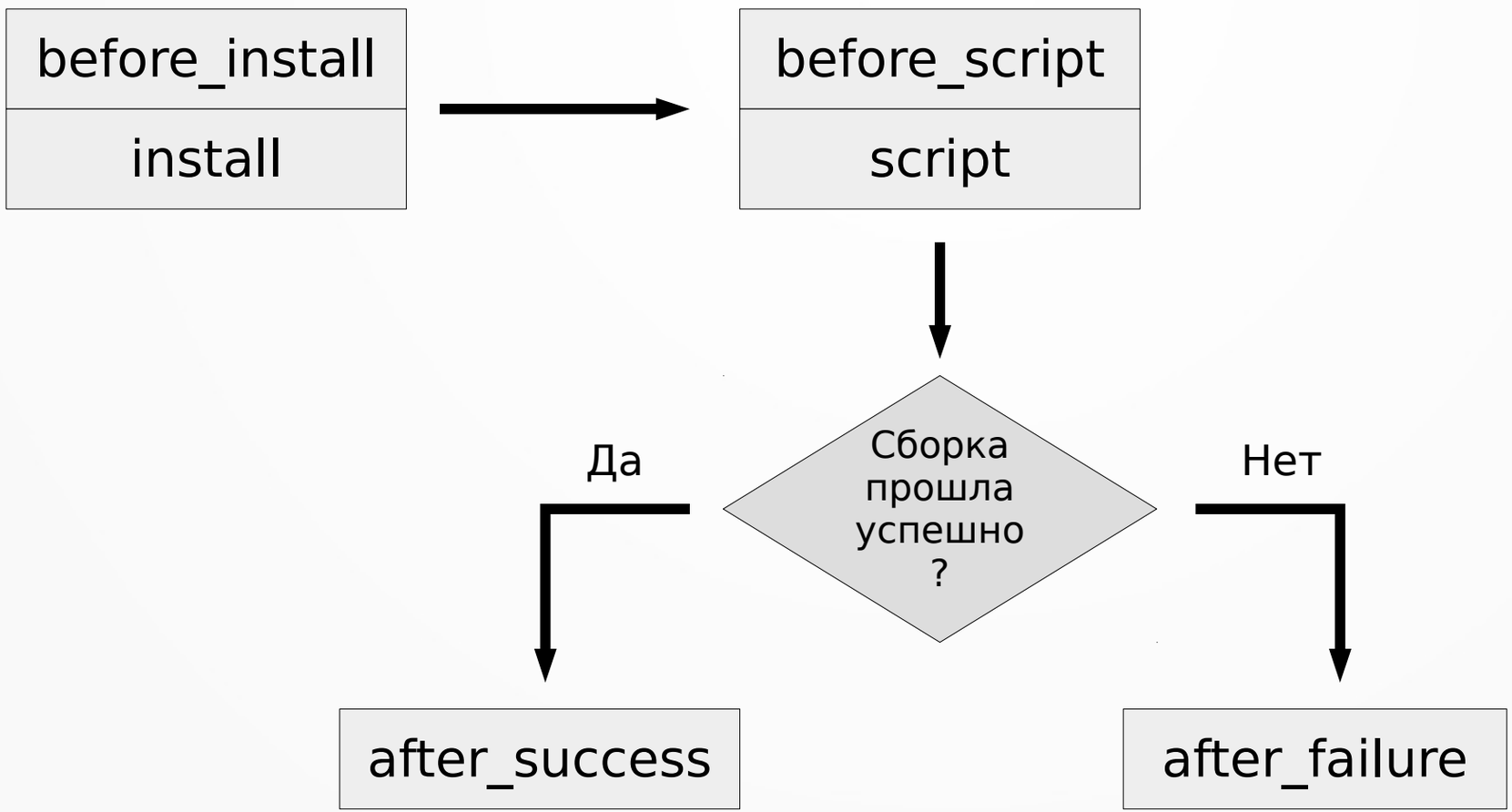
Travis CI Настройка этапов

```
language: cpp
dist: xenial

addons:
  apt:
    update: true
    packages:
      - build-essential
      - cmake
      - googletest
      - ctest
      - jq
      - wget

cache:
    directories:
      - .PVS-Studio/

matrix:
  include:
    - os: linux
      compiler: "gcc"
      env: PVS_ANALYZE=Yes

before_install:
  - travis_retry bash .travis.sh travis_before_install

install:
  - travis_retry bash .travis.sh travis_install

script:
  - bash .travis.sh travis_script

after_success:
  - bash .travis.sh travis_after_success
```

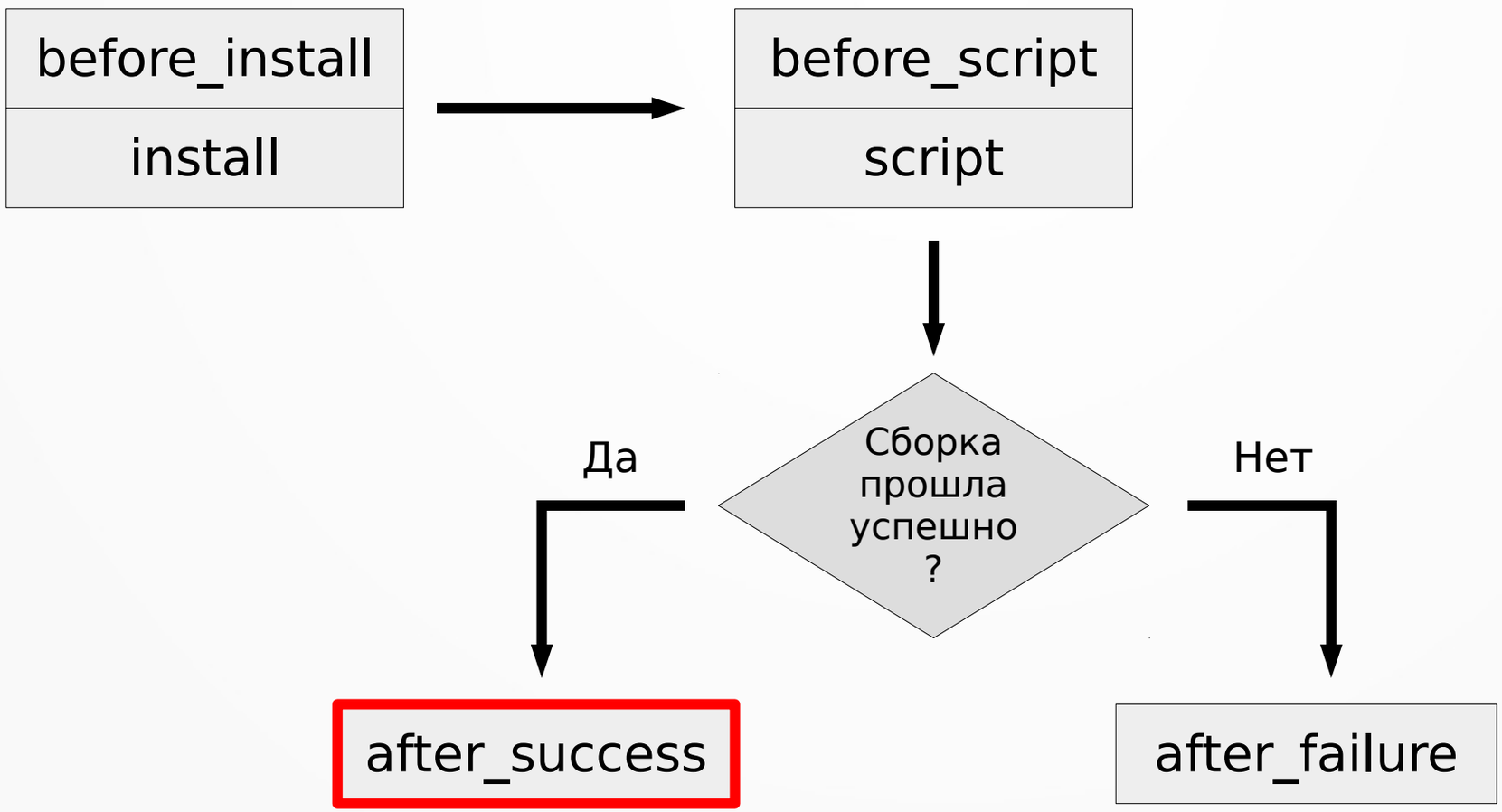
Travis CI Настройка этапов

```
before_install:
  - travis_retry bash .travis.sh travis_before_install

install:
  - travis_retry bash .travis.sh travis_install

script:
  - bash .travis.sh travis_script

after_success:
  - bash .travis.sh travis_after_success
```

Travis CI Job Lifecycle



Travis CI Job Lifecycle



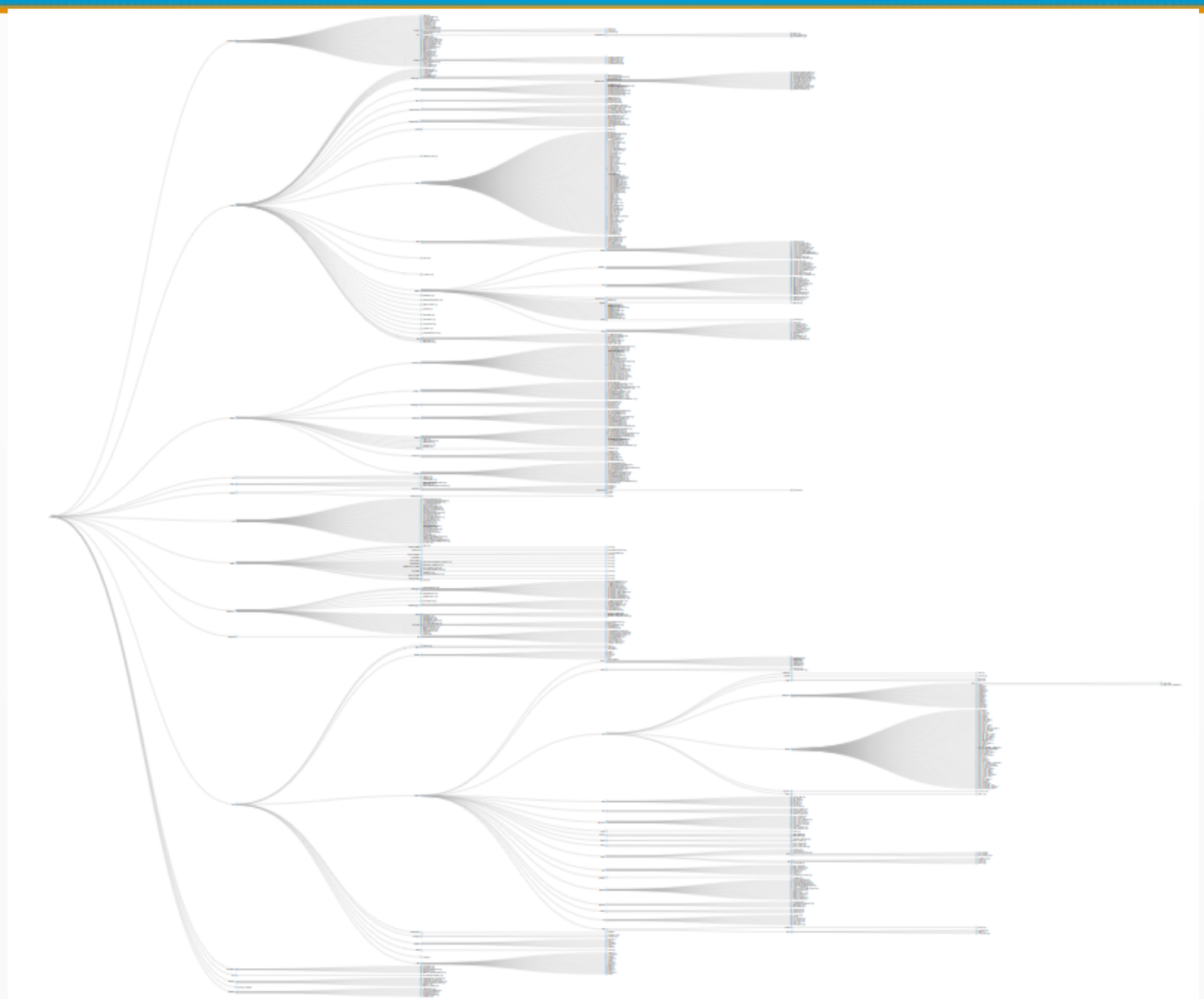
Travis CI Проверка проекта (Unit tests)

```
travis_after_success() {
  ctest -T Test --no-compress-output
}
```

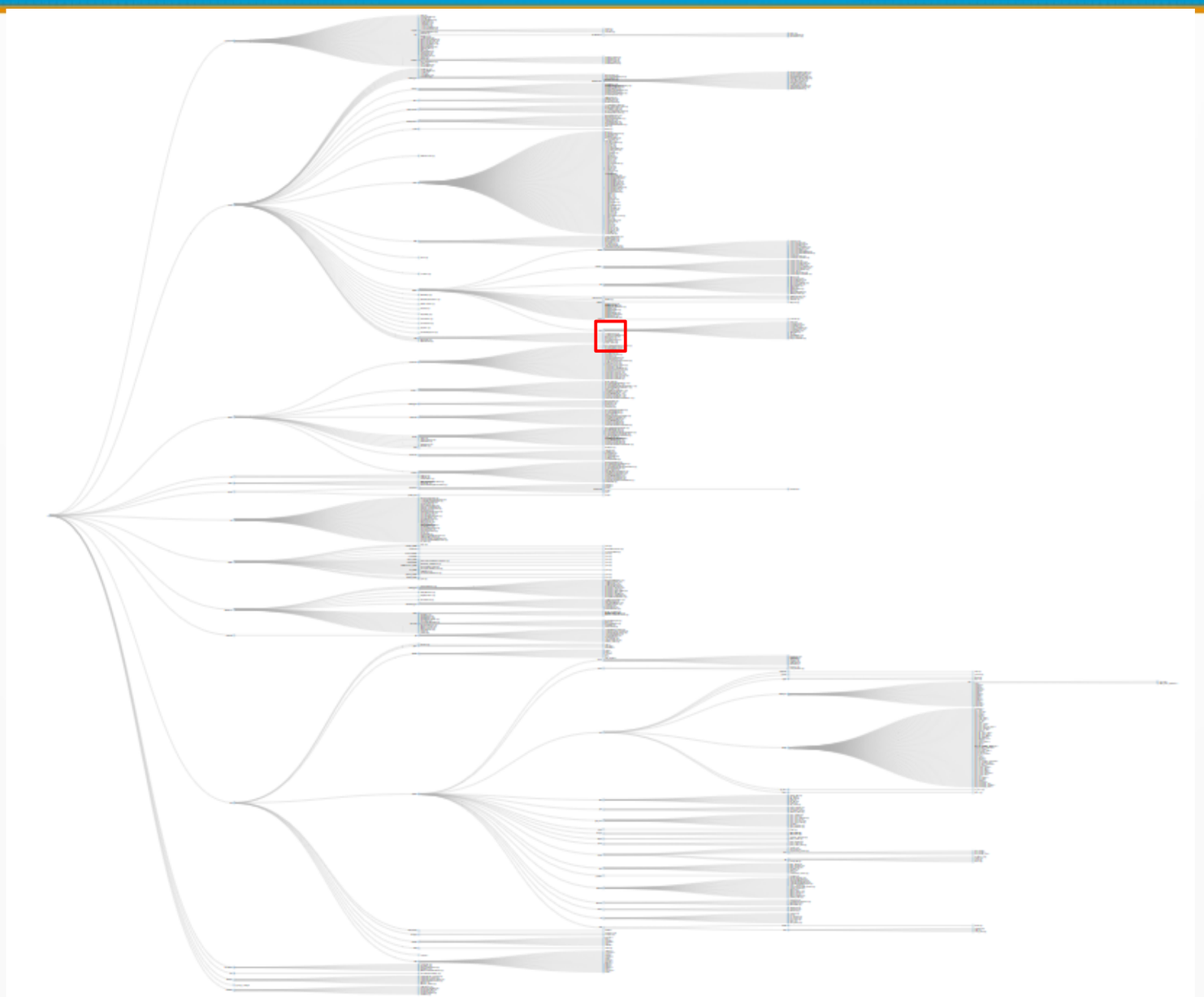
Travis CI Проверка проекта (Статический анализ)

```
travis_after_success() {
  pvs-studio-analyzer credentials $PVS_USERNAME $PVS_KEY

  pvs-studio-analyzer analyze -o PVS-Studio-${CC}.log
  plog-converter -t errorfile PVS-Studio-${CC}.log --cerr -w
}
```

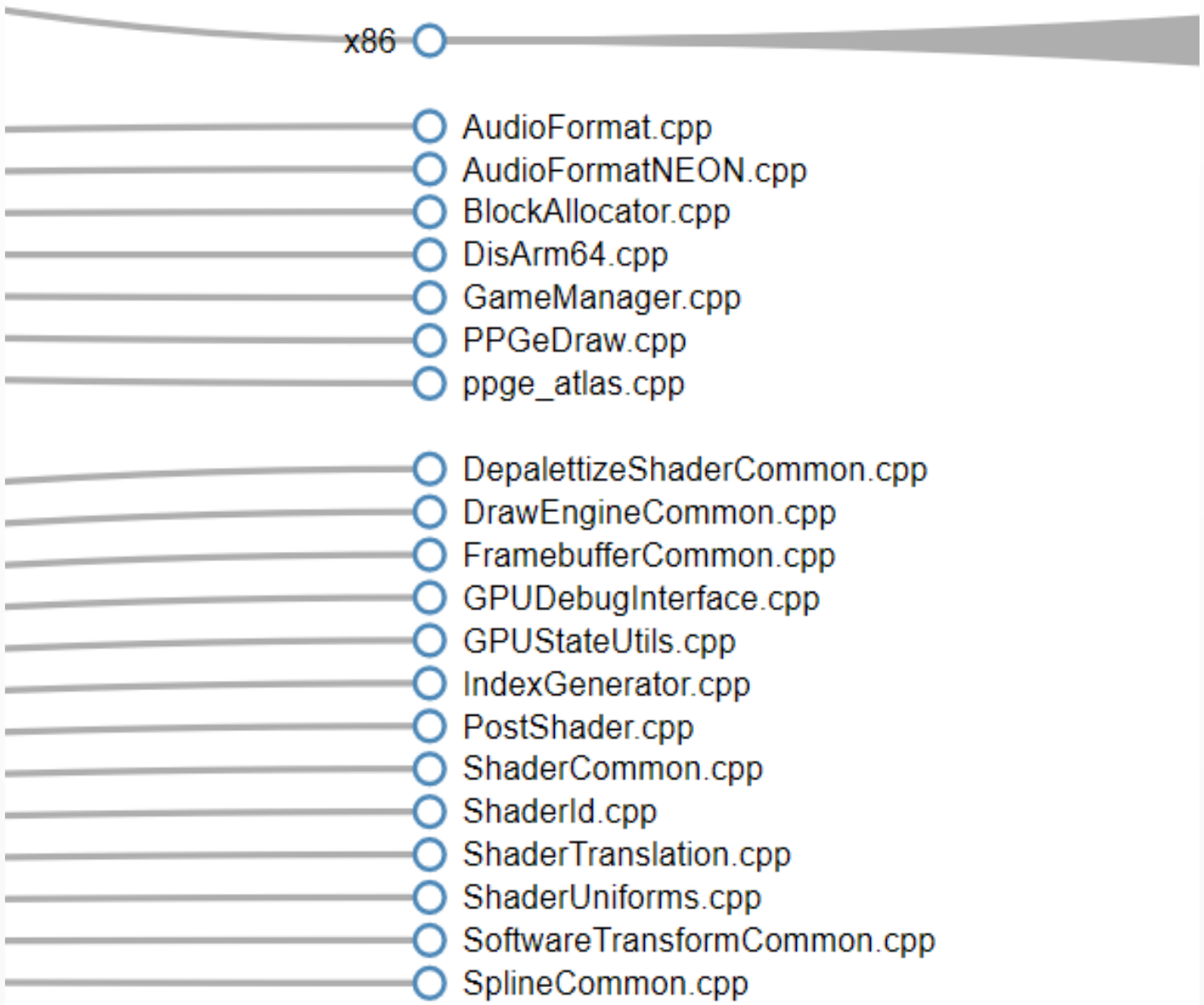
Дерево проекта PPSSPP



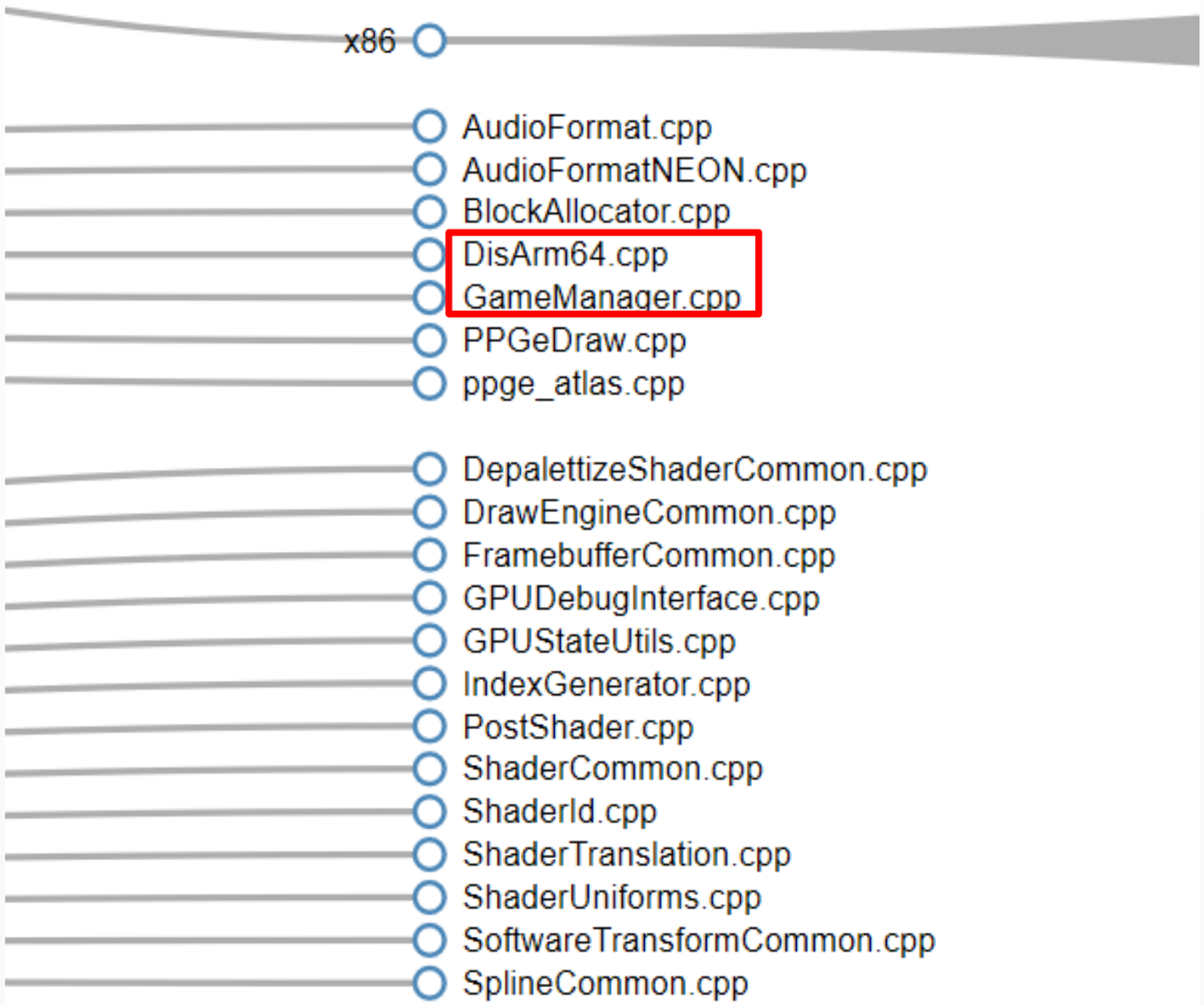
Дерево проекта PPSSPP



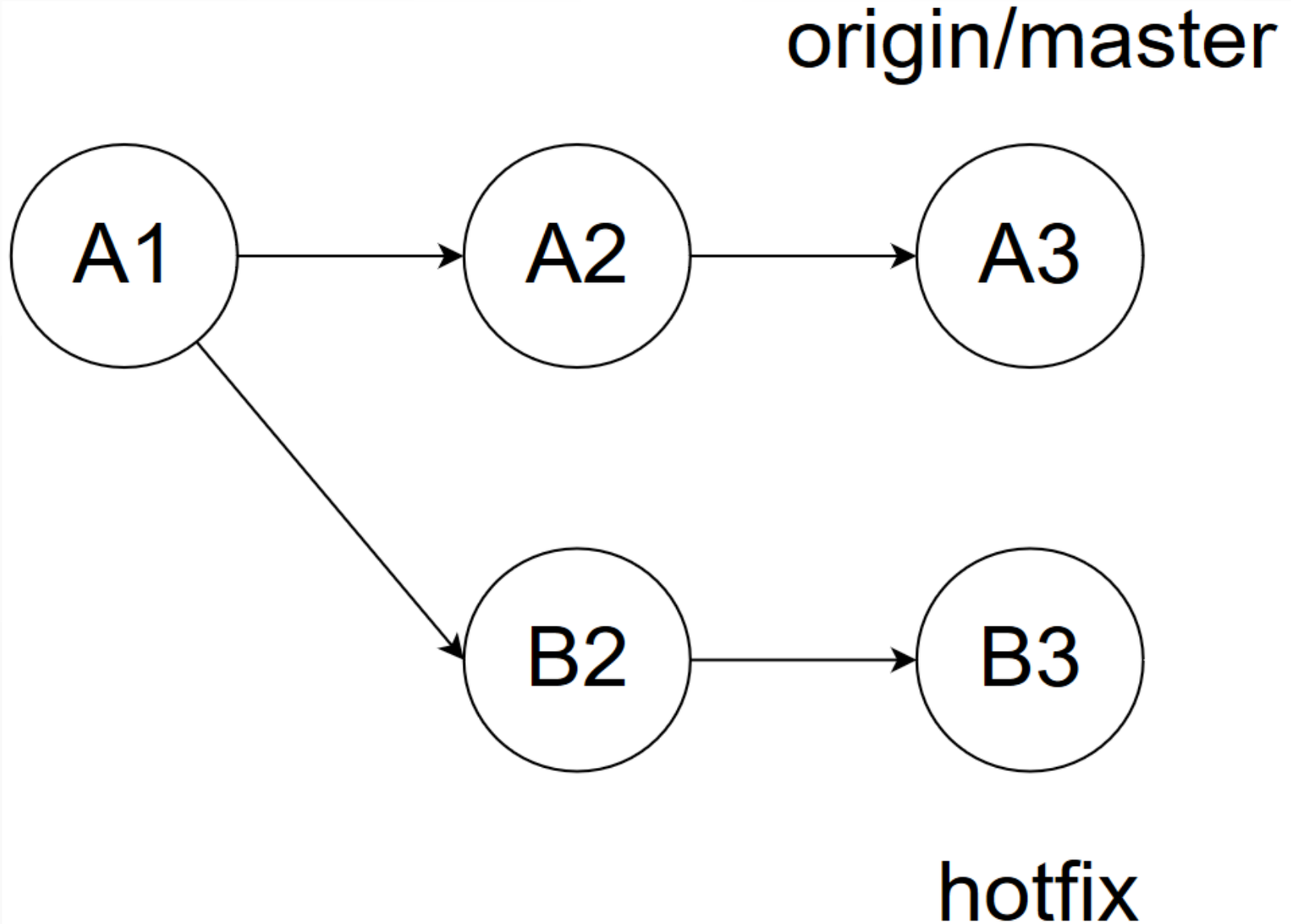
Дерево проекта PPSSPP



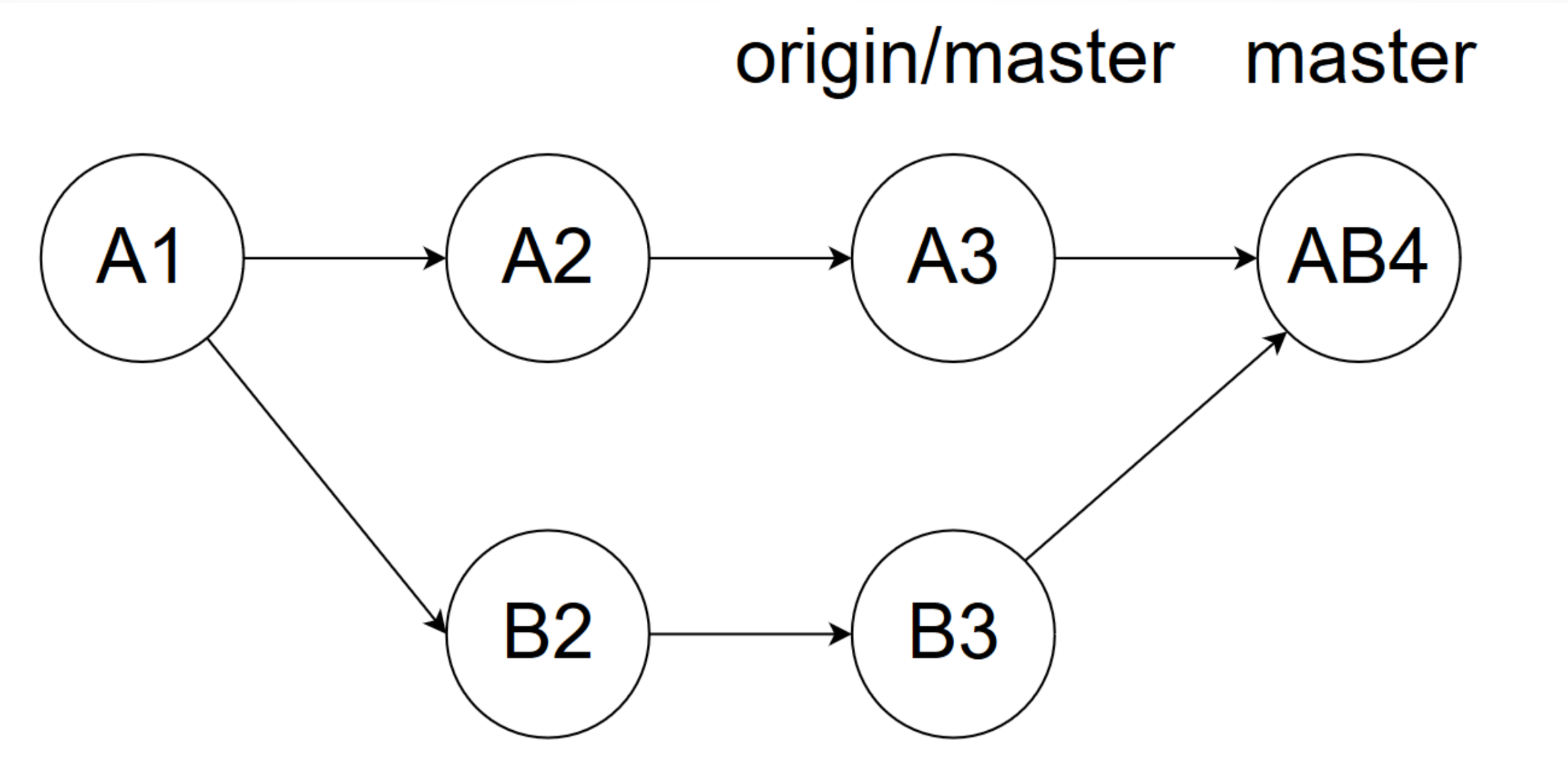
Дерево проекта PPSSPP



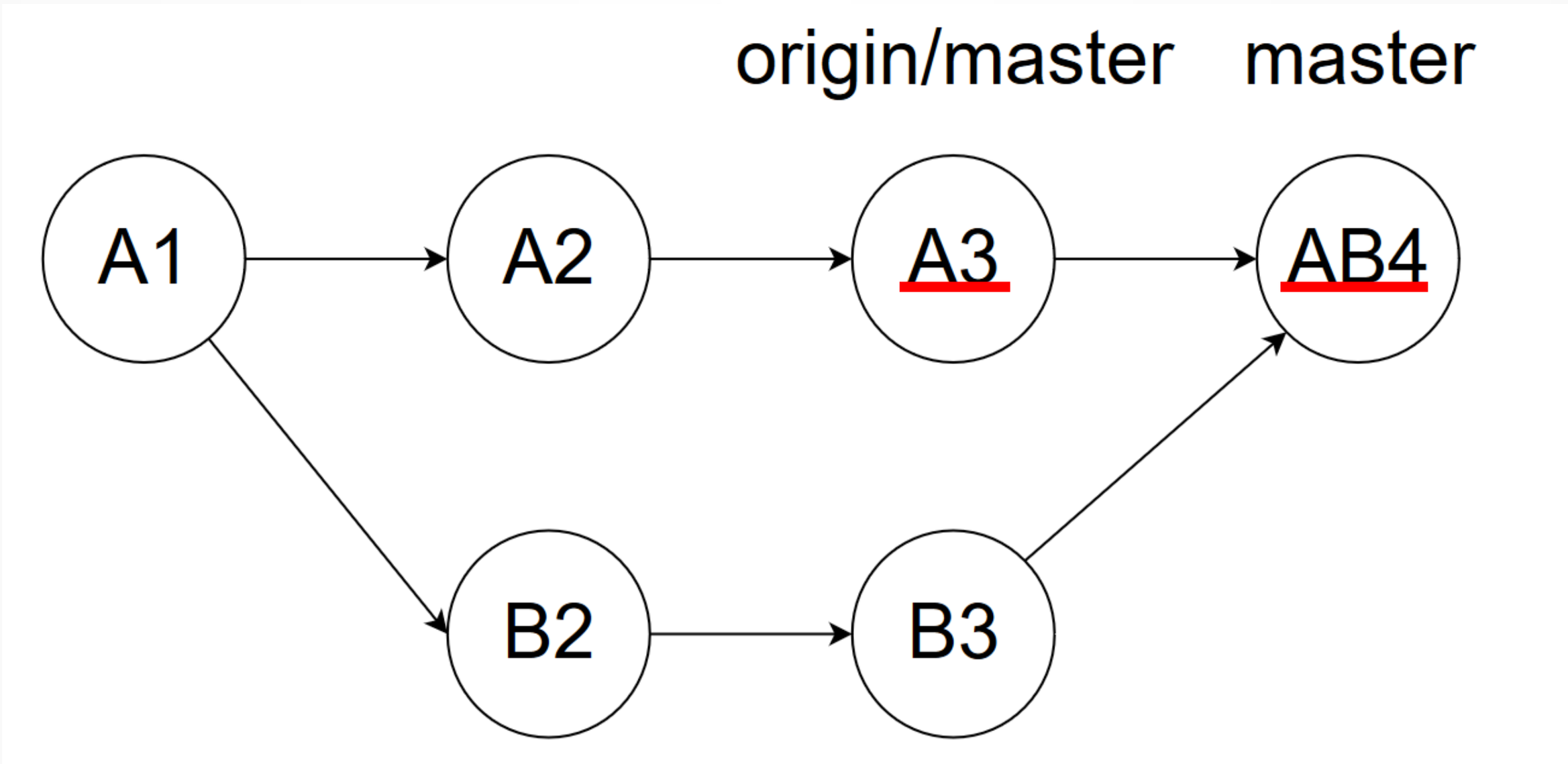
Особенности анализа Pull Request в Travis CI



Особенности анализа Pull Request в Travis CI



Особенности анализа Pull Request в Travis CI



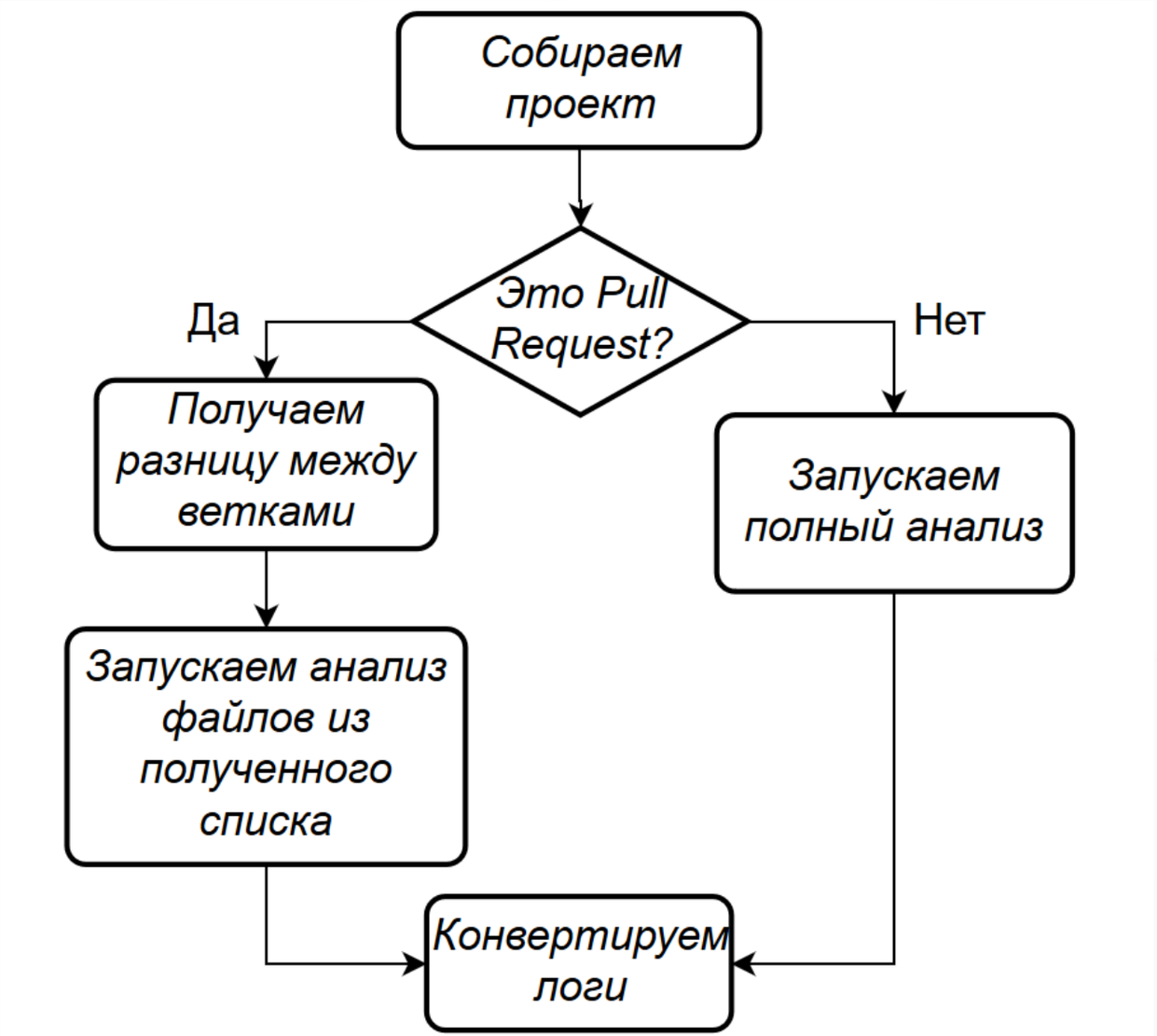
Travis CI Проверка проекта (Статический анализ)

```
travis_after_success() {
  pvs-studio-analyzer credentials $PVS_USERNAME $PVS_KEY

  pvs-studio-analyzer analyze -o PVS-Studio-${CC}.log
  plog-converter -t errorfile PVS-Studio-${CC}.log --cerr -w
}
```

Travis CI Проверка проекта (Статический анализ)

```
travis_after_success() {
  pvs-studio-analyzer credentials $PVS_USERNAME $PVS_KEY

  git diff --name-only origin/HEAD > .pr.list
  pvs-studio-analyzer analyze -o PVS-Studio-${CC}.log \
                              -S .pr.list

  plog-converter -t errorfile PVS-Studio-${CC}.log --cerr -w
}
```

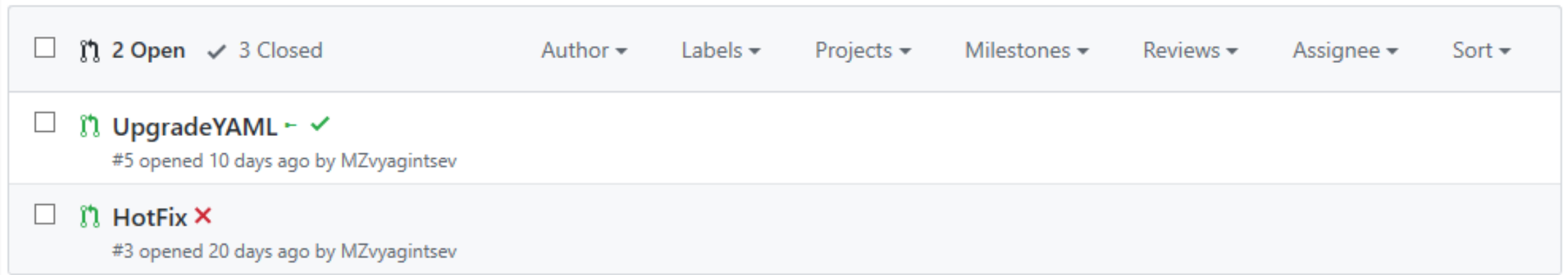
Общий алгоритм работы



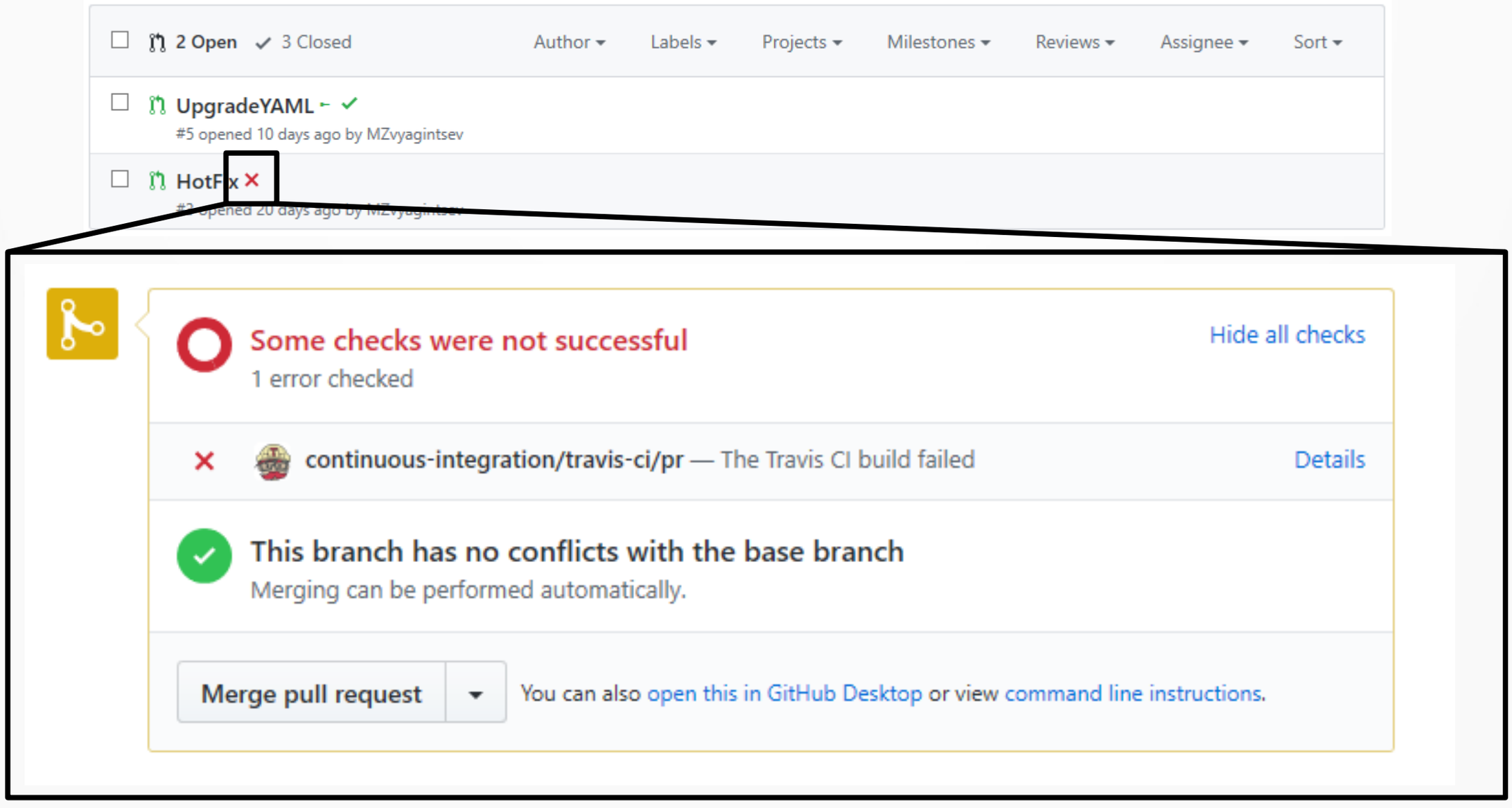
Travis CI Проверка проекта (Статический анализ)

```
travis_after_success() {
  pvs-studio-analyzer credentials $PVS_USERNAME $PVS_KEY

  if [ "$TRAVIS_PULL_REQUEST" != "false" ]; then
    git diff --name-only origin/HEAD > .pr.list
    pvs-studio-analyzer analyze -o PVS-Studio-${CC}.log \
                                -S pr.list
  else
    pvs-studio-analyzer analyze -o PVS-Studio-${CC}.log
  fi

  plog-converter -t errorfile PVS-Studio-${CC}.log --cerr -w
}
```

Travis CI Результат



Travis CI Результат



Итоги

- Чем больше в проекте методов выявления ошибок, тем лучше
- Интеграция в процесс разработки анализаторов кода крайне важна
- CI + различные анализаторы и средства тестирования = увеличение качества кода

Вопросы

<max.post.space@gmail.com>