# Crash репорты Android NDK

Иван Пономарев
Akvelon
ivan.ponomarev@akvelon.com
https://github.com/ivanarh

# Введение

- О чём доклад
- На кого ориентирован
- Сторонние библиотеки с NDK кодом

# Обработка ошибок в Java

```
Thread.UncaughtExceptionHandler handler =
    new Thread.UncaughtExceptionHandler() {
    @Override
    public void uncaughtException(Thread t, Throwable e) {
        e.printStackTrace();
        ...
    }
};
Thread.setDefaultUncaughtExceptionHandler(handler);
```

# Исключения C++

```
void on_terminate() {
    std::exception_ptr eptr = std::current_exception();
    if (eptr) {
        try {
            std::rethrow_exception(eptr);
        } catch (const std::exception &e) {
        }
    }
    abort();
}
…
std::set_terminate(on_terminate);
```

# Прочие C/C++ ошибки

- При работе с памятью (SIGSEGV, SIGBUS)
- При арифметических операциях (SIGFPE)
- Неверная инструкция (SIGILL)
- Вызов abort() (SIGABRT)

# Возможные архитектуры

- In-process
  - Отчёт формируется внутри падающего процесса
  - Ресурсы ограничены
- Out-of-process
  - Отчёт формируется в другом процессе
  - Используется IPC
  - Используется ptrace
  - Реализуем внутри APK в виде Service

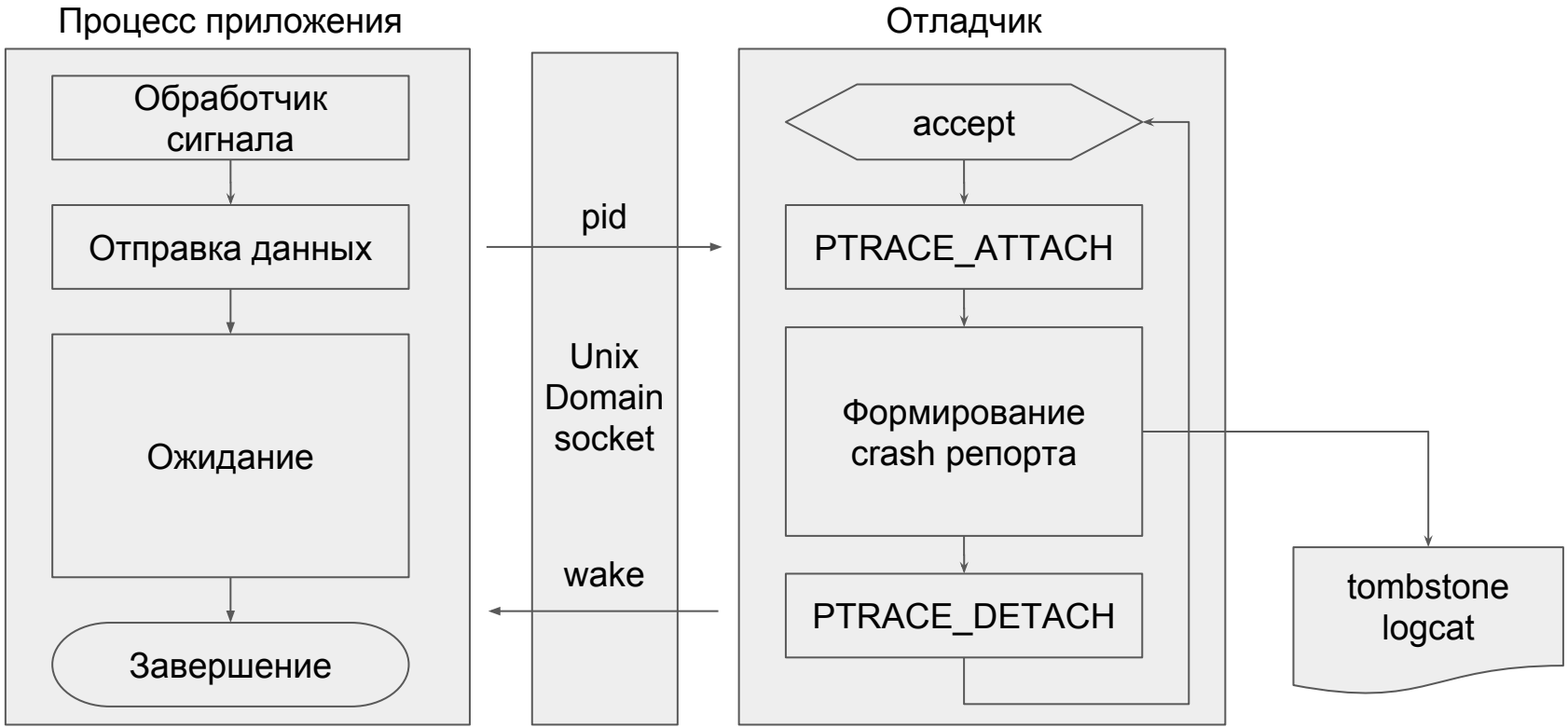
# Стандартные средства Android

# Стандартный отладчик debuggerd

https://source.android.com/devices/tech/debug

- Системный daemon
- Out-of-process
- Требует prctl(PR_SET_DUMPABLE, 1);

# Стандартный отладчик debuggerd



# Tombstone

- Файл с crash репортом
- Хранится 10 последних tombstones
- Недоступен приложению
- Доступен в режиме разработчика

# Tombstone

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
Build fingerprint: 'google/sailfish/sailfish:7.1.2/NJH47D/4045516:user/release-keys'
Revision: '0'
ABI: 'arm'
pid: 14346, tid: 14346, name: r.nativecrashes  >>> com.example.user.nativecrashes <<<
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x0
    r0 00000000  r1 00000000  r2 00000002  r3 00000001
    r4 f072d4d0  r5 ffa1ffb8  r6 ffa1fd60  r7 ffa1fd20
    r8 ffa1ffb8  r9 edf05400  sl ffa1fc70  fp ffa1fc30
    ip ec910fc0  sp ffa1fc2c  lr ec90c8dc  pc ec90c8b0  cpsr 600b0010

backtrace:
    #00 pc 000018b0  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so (doCrash+16)
    #01 pc 000018d8  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so (func2+28)
    #02 pc 00001900  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so (func1+8)
    #03 pc 0000192c  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so
(Java_com_example_user_nativecrashes_MainActivity_crashApp+36)
    #04 pc 000ae399  /system/lib/libart.so (art_quick_generic_jni_trampoline+40)
    #05 pc 000a99c1  /system/lib/libart.so (art_quick_invoke_stub_internal+64)
```

# ndk-stack

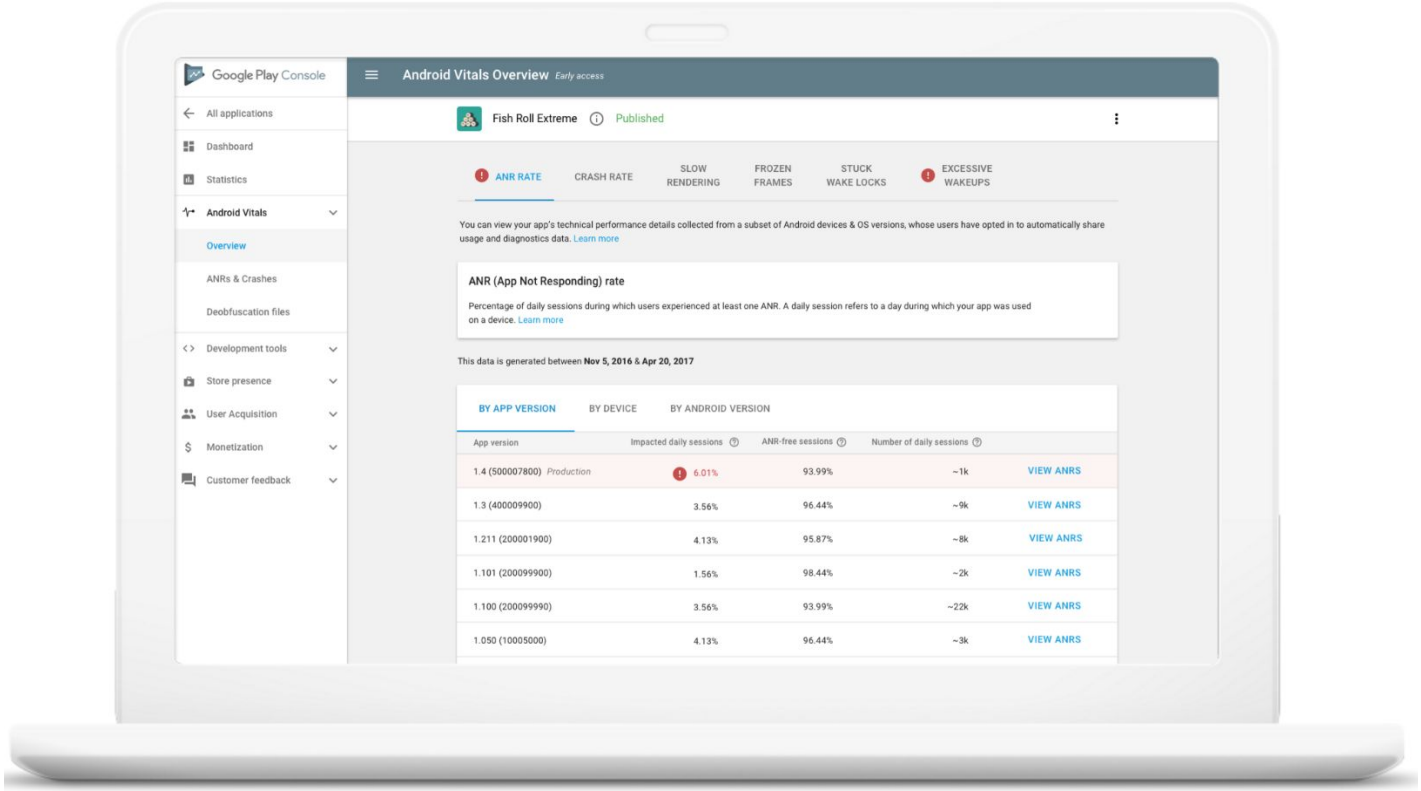
https://developer.android.com/ndk/guides/ndk-stack.html

- Утилита из NDK
- Использует addr2line
- Требует debug символы (DWARF)
- Путь к библиотекам с debug символами: app/build/intermediates/cmake/debug/obj

# ndk-stack

```
********** Crash dump: **********
Build fingerprint: 'google/sailfish/sailfish:7.1.2/NJH47D/4045516:user/release-keys'
pid: 14346, tid: 14346, name: r.nativecrashes  >>> com.example.user.nativecrashes <<<
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x0
Stack frame #00 pc 000018b0  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so
(doCrash+16): Routine doCrash at
/Users/user/src/mobius/nativecrashes/app/src/main/cpp/native-lib.cpp:109
Stack frame #01 pc 000018d8  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so
(func2+28): Routine func2 at
/Users/user/src/mobius/nativecrashes/app/src/main/cpp/native-lib.cpp:117
Stack frame #02 pc 00001900  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so
(func1+8): Routine func1 at
/Users/user/src/mobius/nativecrashes/app/src/main/cpp/native-lib.cpp:125
Stack frame #03 pc 0000192c  /data/app/com.example.user.nativecrashes-1/lib/arm/libnative-lib.so
(Java_com_example_user_nativecrashes_MainActivity_crashApp+36): Routine
Java_com_example_user_nativecrashes_MainActivity_crashApp at
/Users/user/src/mobius/nativecrashes/app/src/main/cpp/native-lib.cpp:132
Stack frame #04 pc 000ae399  /system/lib/libart.so (art_quick_generic_jni_trampoline+40)
Stack frame #05 pc 000a99c1  /system/lib/libart.so (art_quick_invoke_stub_internal+64)
```

# Google play services / Android vitals



# Проблемы

- Недоступно в коде приложения
- Привязка к сервисам Google
- Подтверждение отправки отчёта
- Низкая эффективность

# Обзор существующих решений

# Crashlytics (Fabric)

https://fabric.io/kits/android/crashlytics

- In-process
+ Готовое решение
- Проприетарный

# Bugsnag

https://www.bugsnag.com/

- In-process
+ Готовое решение
+ SDK с открытым кодом
- Неоптимальный алгоритм (полный перебор стека)

# Google Breakpad

https://chromium.googlesource.com/breakpad/breakpad

- in-process или out-of-process?
- minidump в бинарном формате
- Чтение minidump через утилиты
  - Нужны debug символы
  - Требуют linux

\+ Множество платформ, открытый код

\- Сложный к внедрению

\- Нужны символы системных библиотек

# HockeyApp

https://support.hockeyapp.net/kb/client-integration-android/hockeyapp-for-android-ndk-early-access

- Early access с 2015 года
- Использует Google Breakpad
  - Поддержка неполноценная
  - Ручное внедрение Breakpad

# Crasheye

http://www.testplus.us/crasheye

- Китайский
- Использует Google Breakpad

\+ Готовое решение

\- SDK с закрытым кодом

\- Не опубликован в maven

# CoffeeCatch

https://github.com/xroche/coffeecatch

- In-process
- setjmp / longjmp
- NDK backtrace => java.lang.Error

\+ Открытый код

\+ Используется решение для Java ошибок

\- Не работает на Android 7.0 и выше

# Собственный crash reporter

# Требуемые данные:

- **Backtrace**
- Тип ошибки (номер сигнала)
- pid, имя процесса
- tid, имя потока
- Состояние процессора
- Содержимое стека
- Memory map

# Обработка сигналов

```
void on_signal(int signo, siginfo_t *info, void *ctx) {
    …
}
…
struct sigaction sa, old;
memset(&sa, 0, sizeof(sa));
sa.sa_flags = SA_SIGINFO;
sa.sa_sigaction = &on_signal;
sigaction(SIGSEGV, &sa, &old);
```

# Async-signal-safety

- Список безопасных функций http://man7.org/linux/man-pages/man7/signal-safety.7.html
- Воспринимать как рекомендацию

Тип ошибки

```
void on_signal(int signo, siginfo_t *info, void *ctx) {
    ucontext_t *context = (ucontext_t *)ctx;
    int pid = getpid();
    int tid = gettid();
    void *stack = (void*)context->uc_mcontext.arm_sp;
```

Состояние процессора

Содержимое стека

- Имя процесса - /proc/PID/cmdline
- Имя потока - /proc/TID/comm
- Memory map - /proc/PID/maps

# Backtrace?

# Раскрутка стека вызовов (call stack unwinding)

# Исходные данные

Архитектура ARM, 16 регистров R0 - R15.

- R13, SP - адрес вершины стека (stack pointer)
- R14, LR - адрес возврата (link register)
- R15, PC - счётчик команд (program counter, instruction pointer)
- Первый элемент backtrace - значение R15 (PC)
- Второй элемент - R14 (LR)

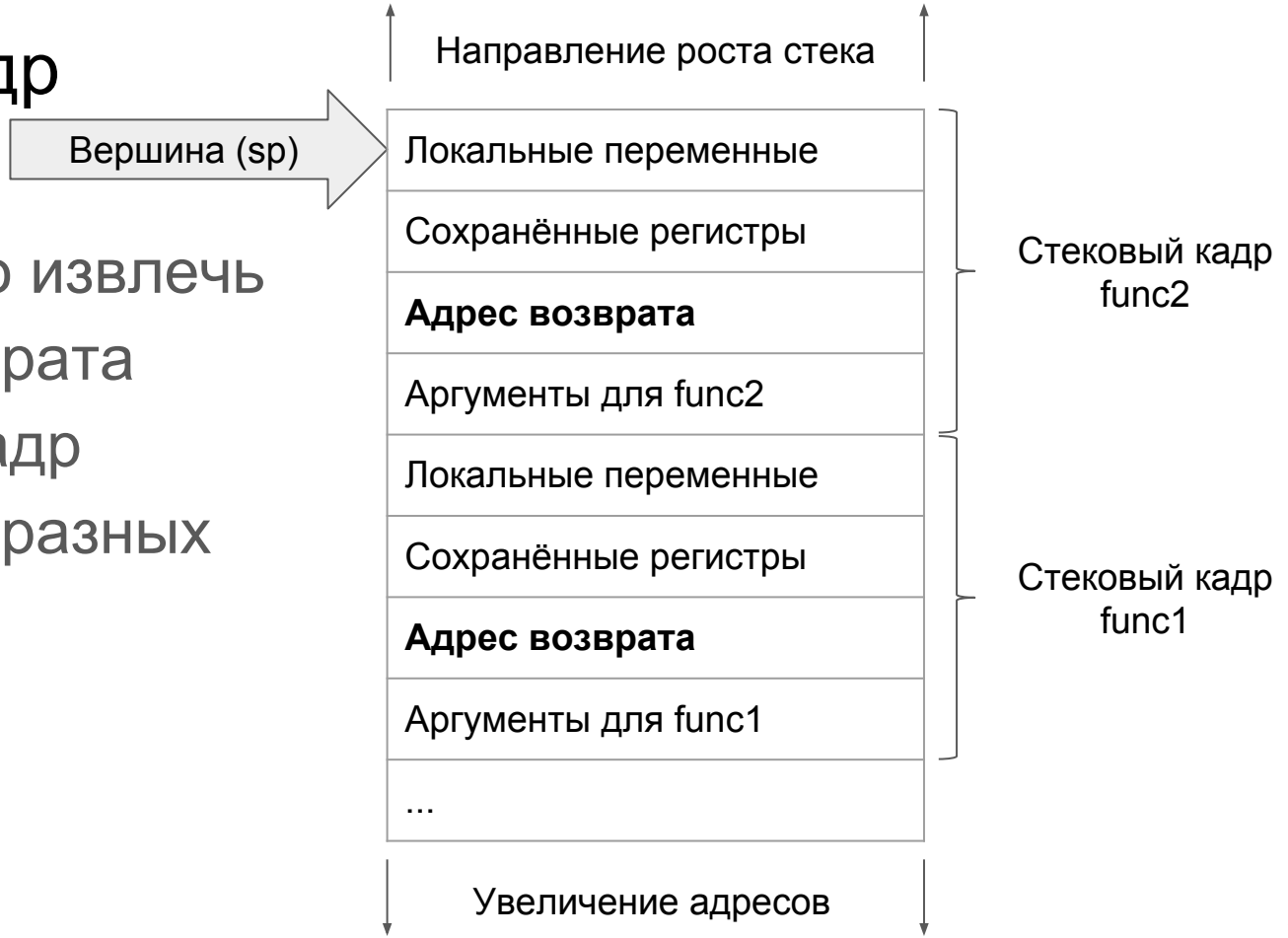
# Имя функции, адрес и смещение?

- Функция dladdr (in-process)
- Анализ memory map и symbol table

```
typedef struct {
    /* Имя файла модуля. */
    const char* dli_fname;
    /* Виртуальный адрес, куда загружен модуль. */
    void* dli_fbase;
    /* Имя функции (имя символа) */
    const char* dli_sname;
    /* Виртуальный адрес начала функции. */
    void* dli_saddr;
} Dl_info;
```

# Стековый кад

- Необходимо извлечь адреса возврата
- Стековый кадр разный для разных функций



# Способы раскрутки стека

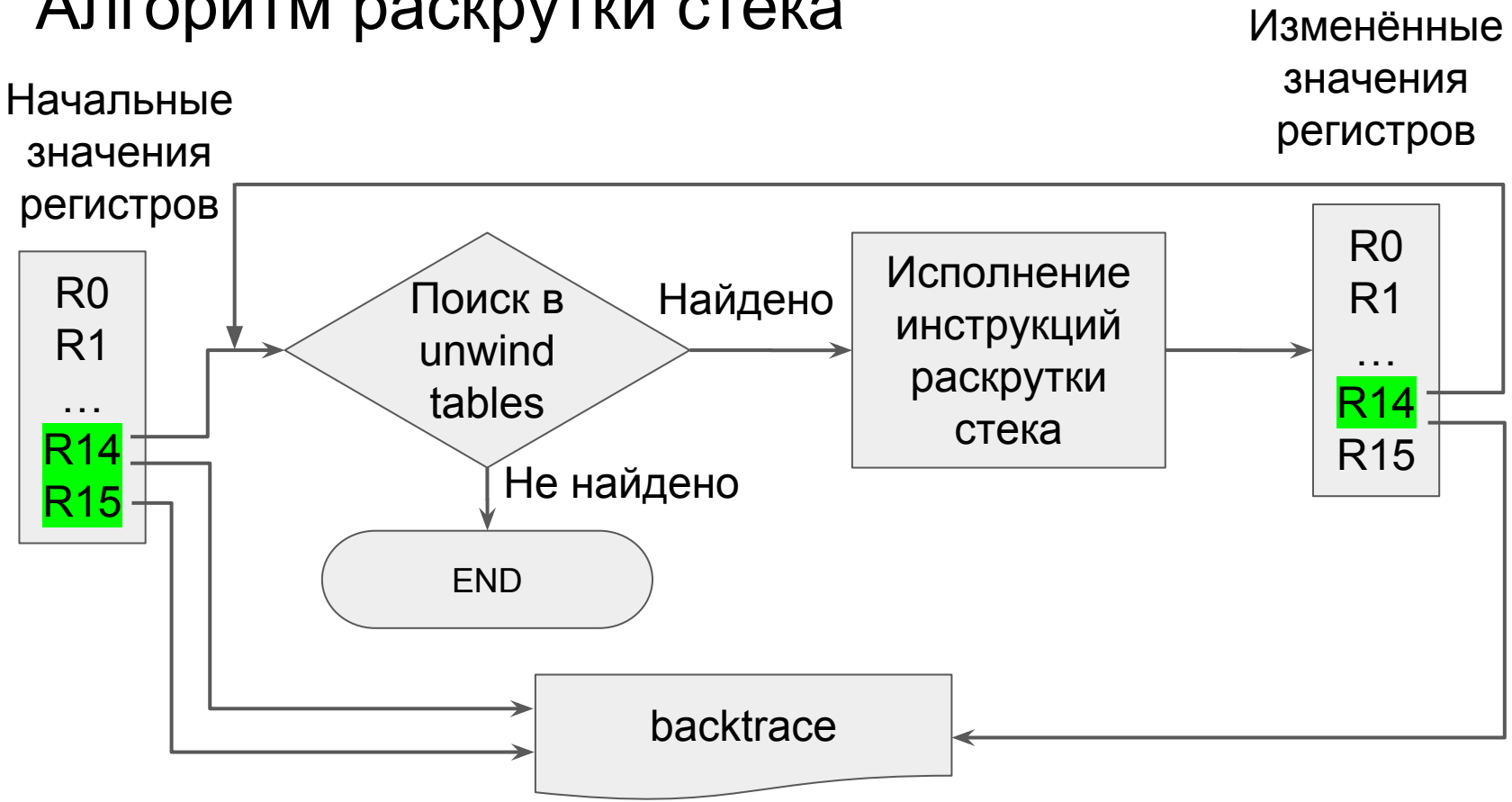
- Полный перебор (Bugsnag)
- Использование отладочных символов DWARF (GDB, LLDB, Google breakpad)
- Использование таблиц раскрутки стека, unwind tables (C++ исключения)

# Таблицы раскрутки стека (unwind tables)

- Секции .ARM.exidx и .ARM.extab
- Один элемент для каждой функции
- Содержат инструкции раскрутки стека
- Не влияют на скорость выполнения

```
$ arm-linux-androideabi-readelf -u lib.so
...
0x21e8 <func2>: @0x39ac
  Compact model index: 1
  0x9b      vsp = r11
  0x40      vsp = vsp - 4
  0x84 0x80 pop {r11, r14}
  0xb0      finish
  0xb0      finish
0x2248 <func1>: @0x39b8
  Compact model index: 1
  0x9b      vsp = r11
  0x41      vsp = vsp - 8
  0x84 0x81 pop {r4, r11, r14}
  0xb0      finish
  0xb0      finish
```

# Алгоритм раскрутки стека



# Флаги компилятора

- Флаг -fexceptions или -funwind-tables **обязателен**
- Флаги оптимизации:
  - -fno-optimize-sibling-calls
  - -fno-inline
- Аккуратнее со strip

```
$ arm-linux-androideabi-readelf -S lib.so
...
[12] .ARM.extab         PROGBITS        0002f060 02f060 0006fc 00   A  0   0  4
[13] .ARM.exidx         ARM_EXIDX       0002f75c 02f75c 000cf0 08  AL 11   0  4
```

# Библиотеки для раскрутки стека

# Стандартная библиотека (cxxabi)

- Unwind tables
- \- Только in-process
- \- Плохо подходит для обработчика сигнала
- \+ Подходит для обработчика C++ исключений

```
_Unwind_Reason_Code callback(_Unwind_Context
*context, void *data)
{
  unwind_data *ud = (unwind_data *)data;
  _Unwind_Word pc = _Unwind_GetIP(context);
  if (pc) {
      Dl_info info;
      dladdr((void *)pc, &info);
      ...
  }
  return _URC_NO_REASON;
}

…
unwind_data unwdata;
_Unwind_Backtrace(callback, &unwdata);
```

# libcorkscrew

- Приватная разделяемая библиотека на Android 4.1 - 4.4
- Возможна статическая линковка
- Язык: C
- Архитектуры: arm, mips, x86
- Unwind tables
- Поддержка in-process и out-of-process (ptrace)
- Версия с патчами для сборки под NDK: https://github.com/ivanarh/libcorkscrew-ndk

# libunwind

- Приватная разделяемая библиотека на Android >= 5.0
- Возможна статическая линковка
- Язык: C
- Архитектуры: все современные
- Unwind tables & DWARF
- In-process и out-of-process (ptrace)
- Имеет обёртку на C++, **libbacktrace**
- Версия с патчами для сборки под NDK: https://github.com/ivanarh/libunwind-ndk

# libunwindstack

- Библиотека из исходников Android >= 8.0
- Возможна статическая линковка
- Язык: C++ 11
- Архитектуры: x86, x86_64, arm, arm64
- Unwind tables & DWARF
- Поддержка in-process и out-of-process (ptrace)
- Версия с патчами для сборки под NDK: https://github.com/ivanarh/libunwindstack-ndk

# Итого, рекомендации

- Использовать out-of-process
- Оптимизировать обработчик сигнала
- Выбор библиотеки по:
  - Архитектурам
  - Языку программирования
  - Размеру кода
- Оптимизировать библиотеку

# Спасибо за внимание

- Примеры кода https://github.com/ivanarh/nativecrashes
- https://source.android.com/devices/tech/debug/
- ARM exception handling ABI http://infocenter.arm.com/help/topic/com.arm.doc.ihi0038b/IHI0038B_ehabi.pdf

Контакты:


- E-mail: ivan.ponomarev@akvelon.com
- Skype: ivan_arh
- Telegram: @ivanarh