Технологии QNX и КПДА в России

Москва, 13 апреля 2017

Применение инструментов статического анализа для поиска программных ошибок

Игорь Рондарев, ООО «СВД Встраиваемые Системы»

СВД Встраиваемые Системы

SWD Software

Содержание

- Статический анализ — что это?
- Инструменты статического анализа
- Пример использования
- Пример интеграции в среду разработки

Статический анализ

- «Стати́ческий ана́лиз ко́да (англ. static code analysis) — анализ программного обеспечения, производимый (в отличие от динамического анализа) без реального выполнения исследуемых программ.»*
  - позволяет выявлять ошибки, трудно поддающиеся локализации с помощью других методик (отладчиков, динамических анализаторов и т.д.)
  - позволяет осуществлять проверку на соответствие стандартам безопасного программирования
  - Помогает по-новому взглянуть на разработанные ранее приложения, в том числе на используемые приёмы и привычки программирования.

* http://ru.wikipedia.org/Статический анализ кода

Пример (базовый статический анализ)

Исходный код (фрагмент):

```
int main(void) {
    char a;
    char array[16];

    /* some long-running-code WITH NO INITIALIZATION */

    printf("result = %d\n", (int)(a + array[8]) );

    return EXIT_SUCCESS;
}
```

Вывод компилятора (**gcc -Wall -Wextra**):

```
static_analysis.c:10: warning: 'a' is used uninitialized in this
function
```

Вывод приложения:

```
result = 42
```

Пример (базовый статический анализ)

Исходный код (фрагмент):

```
int main(void) {
    char a;
    char array[16];

    /* some long-running-code WITH NO INITIALIZATION */

    printf("result = %d\n", (int)(a + array[8]) );

    memset(array, 16, 0xFF); return EXIT_SUCCESS;
}
```

Вывод компилятора (**gcc -Wall -Wextra**):

```
static_analysis.c:10: warning: 'a' is used uninitialized in this
function
```

Вывод приложения:

```
result = 42
Segmentation fault
```

Пример (базовый статический анализ)

Исходный код (фрагмент):

```
int main(void) {
    char a;
    char array[16];

    /* some long-running-code WITH NO INITIALIZATION */

    printf("result = %d\n", (int)(a + array[8]) );

    memset(array, 0xFF, 16); return EXIT_SUCCESS;/* Correct! */
}
```

Вывод компилятора (**gcc -Wall -Wextra**):

```
static_analysis.c:10: warning: 'a' is used uninitialized in this
function
```

Вывод приложения:

```
result = 42
```

Пример (базовый статический анализ)

- Алгоритмические ошибки
  - «*off-by-one*»
    - `char a[4]; int i; for(i=0; i<=4; i++) {…}`
- Синтаксические ошибки
  - `memset (dst, val, len) <=> memset (dst, len, val)`
- **Смысловые ошибки**
  - Ошибки «Copy-Paste»
    - `if(obj.x == x || obj.y == y || obj.z == y) {...}`
  - «Похожие» функции
    - Класс `std::string`
      - `str.empty()` вместо `str.clear()`
- и т.д.

Статические анализаторы (C/C++)

- Свободные/открытые:
  - **cppcheck**, **cpplint**, **splint**, **Frama-C** и т. д.
- Коммерческие:
  - **Parasoft C/C++test, Coverity**, **LDRA TestBed**, **PC-Lint**, **PVS-Studio** и т.д.

- Статический анализатор **cppcheck**
  - Продукт с открытым исходным кодом
  - Гибкая настройка
    - Используемые стандарты языка
      - **C99, C++11** и т.д.
    - Используемые библиотеки и функции
      - **POSIX, Qt, GTK, SDL** и т. д.
    - Учёт особенностей анализируемых проектов
      - **Создание пользовательских правил**
  - Возможность запуска в **многопоточном** режиме
    - Сокращает время анализа больших проектов

cppcheck (продолжение)

- Варианты использования
  - **CLI (Command Line Interface)**
    - Запуск для анализа проектов или индивидуальных файлов
    - Автоматизированный запуск для построения отчётов
  - **GUI (Graphical User Interface)**
    - cppcheck-gui (входит в состав cppcheck)
    - QNX Momentics IDE
    - Qt Creator
- Документация
  - http://cppcheck.sourceforge.net/manual.pdf

- Анализатор **cppcheck**
  - Установка на инструментальную систему
    - Windows
      - Готовые сборки (32-bit/64-bit): http://cppcheck.sourceforge.net/
    - GNU/Linux
      - Установка из репозитория дистрибутива
        - Debian: **$ sudo apt install cppcheck**
      - Сборка из исходных кодов
        - https://github.com/danmar/cppcheck/releases/
  - Базовая настройка
    - Стандарты языка и особенности ОС
    - Формат и содержимое вывода
  - Запуск анализатора
  - Использование **cppcheck-gui** для просмотра результатов анализа

---

* рекомендуемая версия QNX Momentics IDE - 5.0 и выше

Практическая часть (продолжение)

- Средства интеграции в QNX Momentics IDE* (модуль **cppcheclipse**)
  - Установка
  - Базовая настройка
  - Использование
    - импорт проекта, запуск анализатора и просмотр результатов анализа

* рекомендуемая версия QNX Momentics IDE - 5.0 и выше

Итоги

- Статический анализатор **cppcheck**
  - Проект активно развивается
    - https://sourceforge.net/p/cppcheck/news/
  - Поддерживаются различные инструментальные платформы
    - **Windows-based**
    - **Linux-based**
  - Развитый пользовательский интерфейс
    - **Интеграция с IDE**
  - **Расширяемость**, в т.ч. возможность создания дополнительных правил анализа
  - Возможность использования в **системах автоматизированной сборки и тестирования ПО**
  - Подробная документация и поддержка со стороны сообщества

Спасибо за внимание

Игорь Рондарев

инженер-программист

ООО «СВД Встраиваемые Системы»

тел.: +7 (812) 346-8956
факс: +7 (812) 346-8953

http://www.kpda.ru | http://forum.kpda.ru |
http://www.swd.ru

СВД Встраиваемые Системы

SWD Software