Технологии QNX и КПДА в России

Москва, 25 апреля 2018

«Расширенные возможности инструментов статического анализа кода»

Александр Покид, СВД Встраиваемые Системы

СВД Встраиваемые Системы

SWD Software

Что такое статический анализ?

Статический анализ программы – это анализ, который выполняется без фактического выполнения программы.

Задачи:

- Выявление ошибок в программах;
- Рекомендации по оформлению кода;
- Подсчет метрик.

Статический анализ и динамический анализ

Статический анализ	Динамический анализ
• Более полное покрытие ветвей потока выполнения программы • Как правило, менее требователен к вычислительным ресурсам • Не зависит от используемого компилятора или среды	• Отсутствие ложных срабатываний • Не требует исходного кода • Более эффективен в диагностике утечек памяти и параллельных ошибок

Типы обнаруживаемых ошибок

Ошибки, обнаруживаемые статическими анализаторами:

- Неопределённое поведение;
- Нарушение алгоритма пользования библиотекой;
- Сценарии, приводящие к недокументированному поведению (gets, strcpy);
- Переполнение буфера;
- Сценарии, мешающие кроссплатформенности;
- Ошибки форматных строк;
- Неиспользуемые переменные;
- Другие.

Примеры выявляемых ошибок

Исходный код:

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

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

   printf("result = %d\n“, a + array[8]);
   return EXIT_SUCCESS;
}
```

Вывод компилятора (gcc –Wall –Wextra):

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

Примеры выявляемых ошибок

Исходный код:

```
void func( int itemNum ) {
  ...
  item_t item;
  ...
  memset(&item, 0, sizeof( &item) );
  ...
}
```

Вывод статического анализатора:

```
(warning, inconclusive) Size of pointer
'struct_ptr' used instead of size of its data.
```

Инструменты статического анализа

Название	Лицензия	Windows	Linux	Нативная поддержка Momentics IDE	Последнее обновление*
cppcheck	GPLv3	+	+	Есть	2017.11.15
cpplint	BSD	+	+	Нет	2017.11.10
splint	GPL	+	+	Нет	2014.11.14
Frama-C	LGPL	+	+	Есть	2017.05.01
Coverity	Проприетарная	+	+	Есть	2017.07.01
PC-Lint	Проприетарная	+	-	Есть	
PVS-Studio	Проприетарная	+	+	Нет	2017.11.14

СВД Встраиваемые Системы

* - даты указаны на 2017.11.17

SWD Software

Преимущества анализатора cppcheck

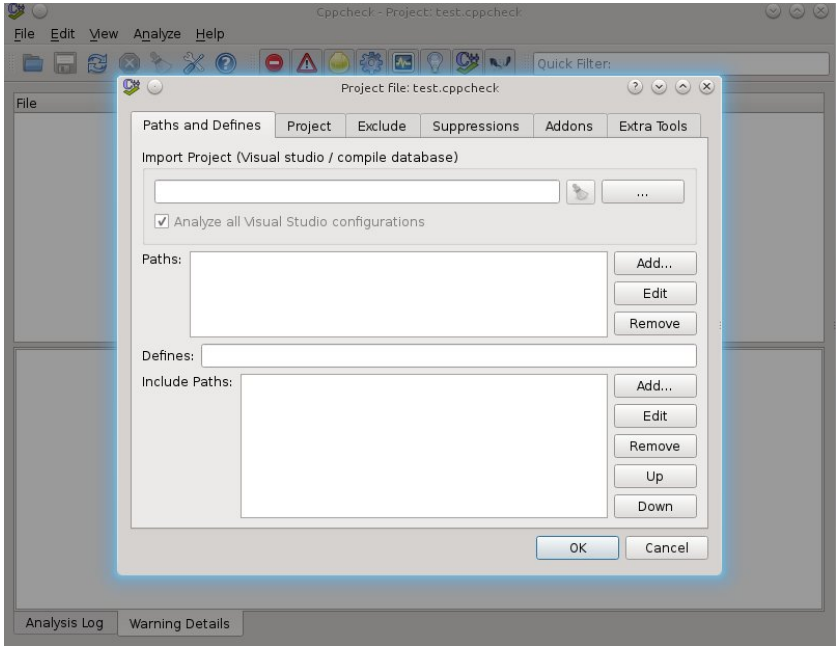
- Кроссплатформенность;
- Гибкость настроек сканирования;
- Интеграция с большим количеством IDE;
- Поддержка со стороны сообщества;
- Наличие CLI и GUI версии;
- Возможность настройки формата выводимых данных.

Использование cppcheck

CLI-версия:

```
# cppcheck <path>
Checking path/file1.cpp...
1/2 files checked 50% done
Checking path/file2.cpp...
2/2 files checked 100% done
```

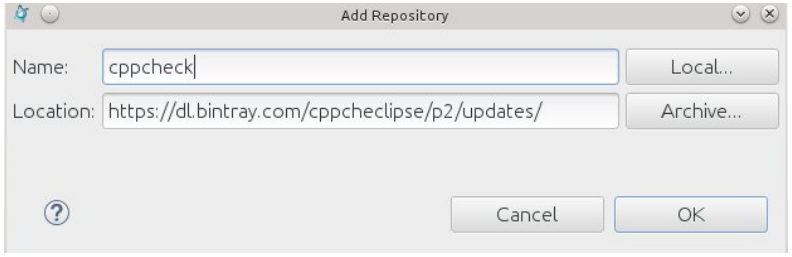
GUI-версия:



Cppcheck интеграция в Momentics IDE

- **Установка (Momentics7)** :

1. Добавление репозитория:
   “Help” -> “Install new software” -> “Add”
   https://dl.bintray.com/cppcheclipse/p2/updates/
2. Установка
3. Настройка и конфигурирование:
   “Window” -> “Preferences” -> “C/C++” -> “cppcheclipse”



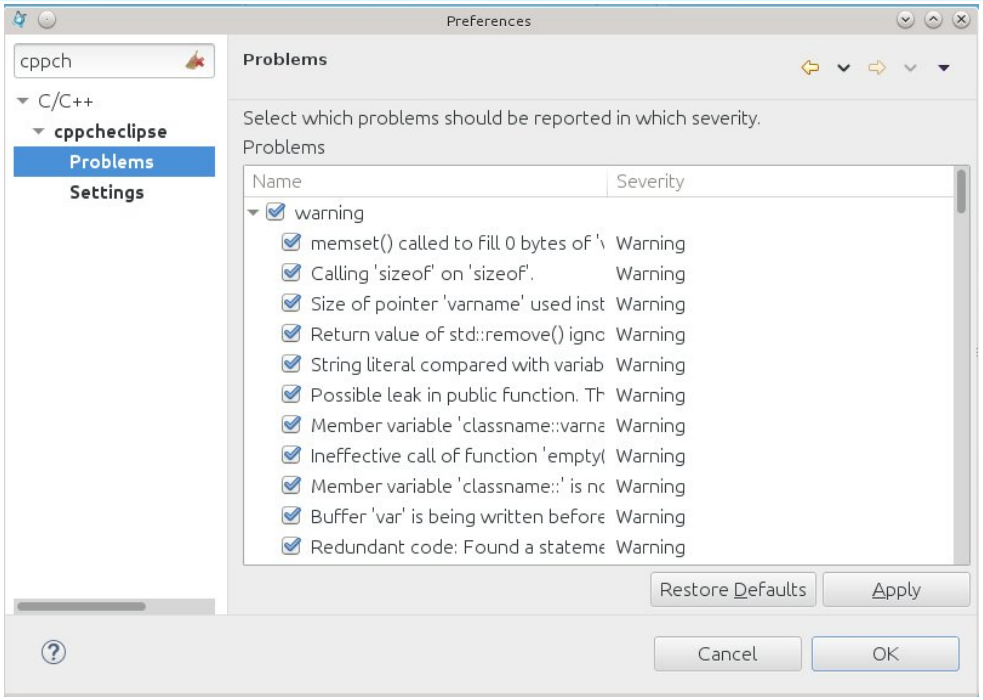
SWD Software

Конфигурирование cppcheck

- Поддерживаемые **категории проверок** (опция --enable=<category> для CLI):
  - error;
  - warning;
  - style;
  - performance;
  - portability;
  - information;
  - unusedFunction (только для однопоточного режима);
  - missingInclude.
- Поддерживаемые **платформы** (опция --platform для CLI):
  1. unix32, unix64 – Unix-подобные платформы
  2. win32, win64 – операционная система Windows
- **Стандарты** (опция --std для CLI): posix, c89, c99, c11, c++03, c++11

Фильтрация сообщений об ошибках

Фильтрация сообщений



- GUI:

“Windows” -> “Preferences”

-> “cppcheclipse” ->“problems”

- CLI:
  1. Опция командной строки : --suppress=<исключение>
  2. Файл с исключениями:
     --supressions-list=<файл с исключениями>

Формат исключений:

<идентификатор>[:<файл>:[<номер строки>]]

Пример: --supress=unreadVariable:myfile.c:42

SWD Software

Изменение формата вывода

- Использование формата вывода компилятора (опция --template <gcc|vs>)

- Использование собственного формата вывода (опция --template <шаблон>)

Спецификаторы шаблона:

- file – имя файла;
- id – идентификатор сообщения;
- line – номер строки;
- message – сообщение ;
- severity – тип/ранг сообщения.

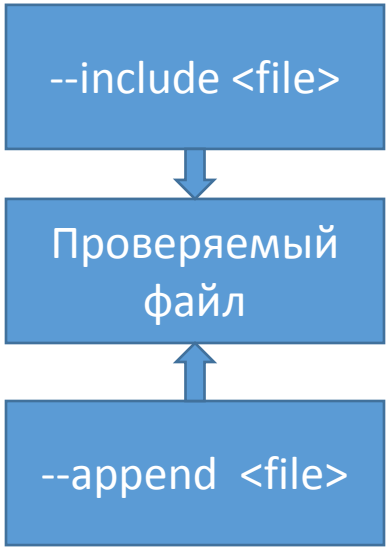
Пример: `--template='{id} {file}:{line} {message}'`

Подключение заголовочных файлов

- Указание пути для поиска заголовочных файлов и определение макросов через опции командной строки.

Пример: `cppcheck -I/usr/local/include/ -DMY_MACRO`

- Самостоятельная реализация недостающих макросов и функций:
  1. С помощью добавления файлов в начало каждого проекта (опция --include);
  2. С помощью добавления файлов в конец каждого проекта (опция --append).



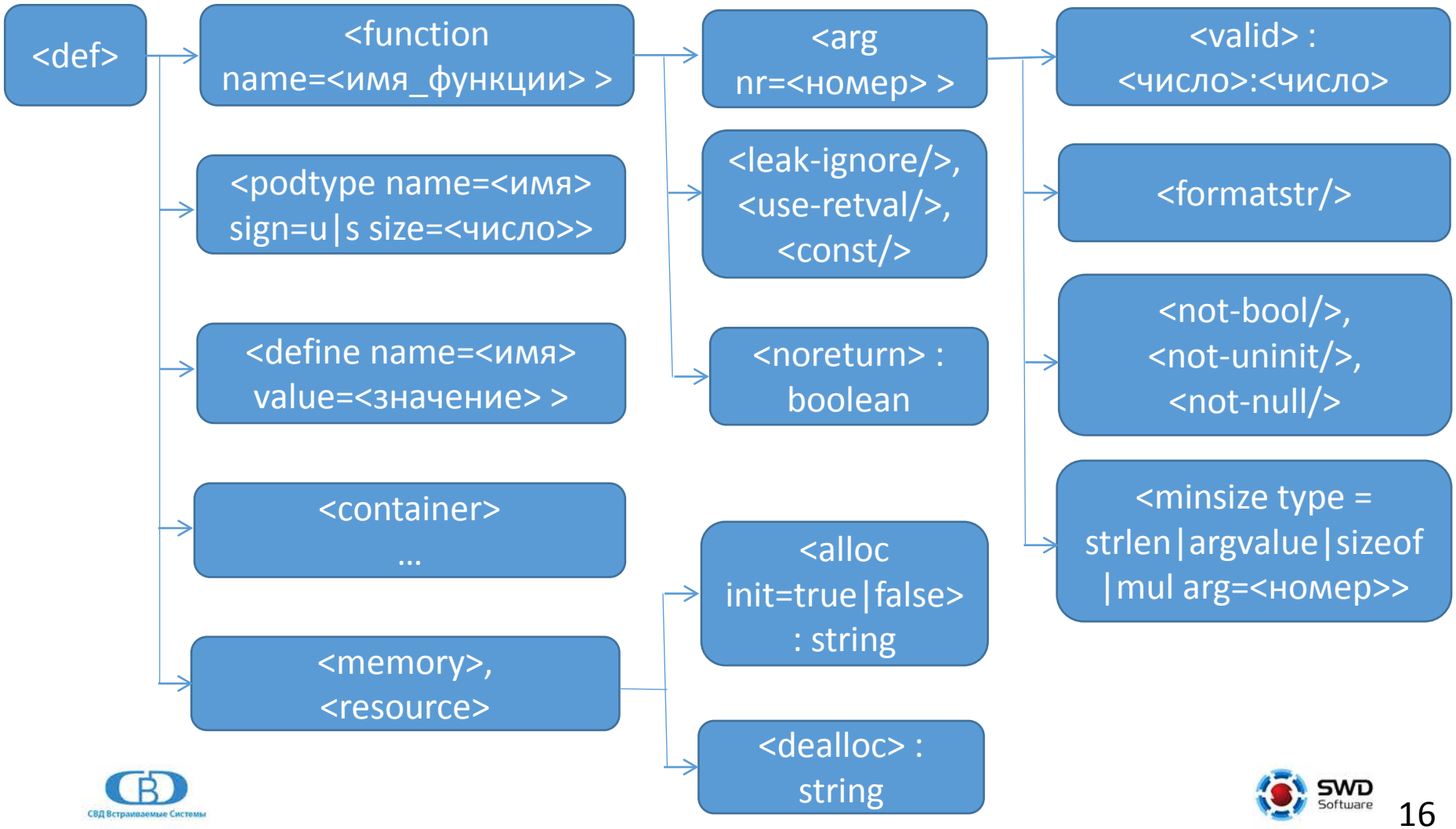
Настройка на использование специфичных библиотек

- Поддерживаемые библиотеки (опция --library=<библиотека или путь к ней>):
  - Gtk;
  - Qt;
  - Windows;
  - Posix;
  - Glibc.

- Описания библиотек хранят информацию о:
  - Списке функций выделения памяти;
  - Списке функций прерывающих выполнение;
  - Информацию об аргументах функций;
  - Списке функций для которых необходимо выполнять проверку строки-формата.

Структура файла-дескриптора библиотеки

Формат XML:



Пример файла-дескриптора библиотеки

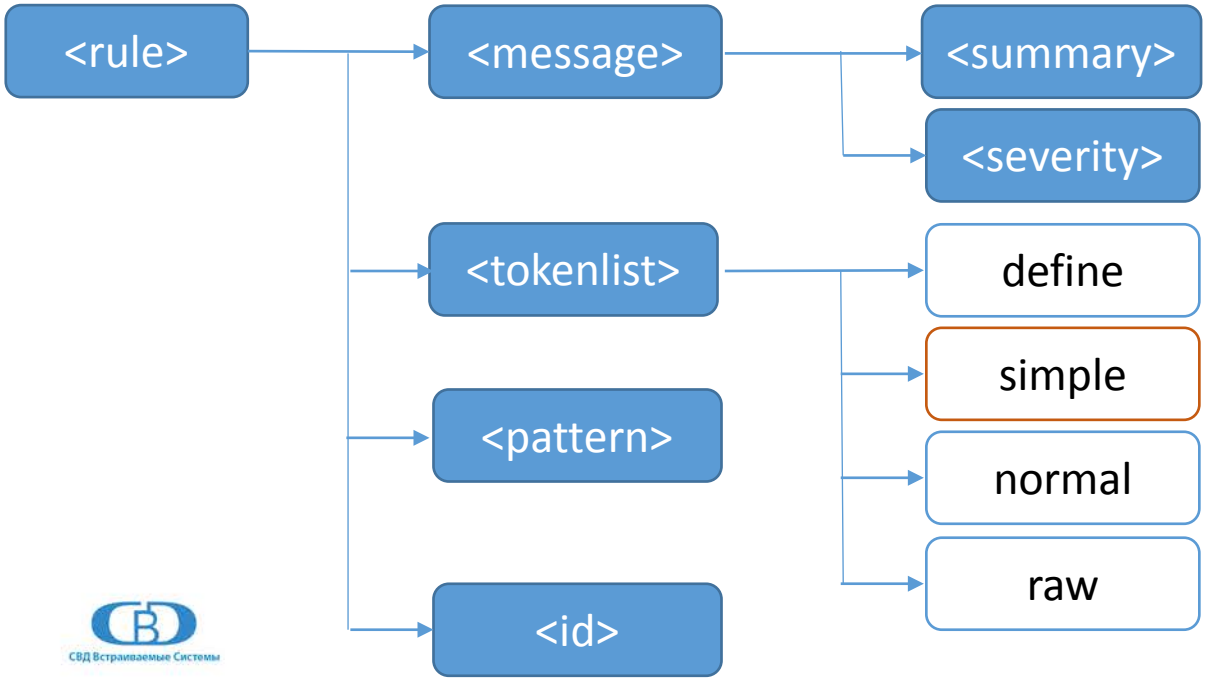
Пример файла:

```
<?xml version="1.0"?>
    <def>
       <function name="CopyMemory">
           <arg nr="1">
               <not-null/>
           </arg>
           <arg nr="2">
               <not-uninit/>
           </arg>
           <arg nr="3"/>
       </function>
     </def>
```

Добавление правил

- Способы добавления правила:
  - Опция командной строки (--rule=<регулярное выражение>)
  - XML файл с правилами (опция --rule-file=<путь к файлу>)

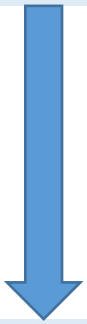
Формат XML:

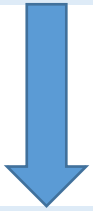


Препроцессинг исходных файлов

```
for (int i = 0; i < 10; i++)
    if(i % 2)
        printf("%d\n", i);
```

```
int i;
for(i = 0; i < 10; i++) {
    if(i % 2)
        printf("%d\n", i);
}
```





```
int i ; for ( i = 0 ; i < 10 ; i ++ ) { if ( i % 2 ) { printf
( "%d\n" , i ) ; } }
```

Добавление правил

Пример файла:

```
<?xml version="1.0"?>
<rule>
    <pattern>if \( .* \) { .* }</pattern>
    <message>
        <severity>style</severity>
        <summary>Simple error rule detected!</summary>
    </message>
</rule>
```

Ссылки и материалы

- Моисеев М. Методы анализа и обеспечения качества ПО. СПб: Санкт-Петербургский Политехнический Университет. Курс лекций, 2011.
- Cppcheck 1.8 User Manual.
- Рондарев И. Выявление программных ошибок при помощи инструментов статического анализа. 2016
- Статический анализ кода // Intel Software URL: https://software.intel.com/ru-ru/articles/Static-code-analysis-ru.
- Anders Moller and Michael I. Schwartzbach. Static Program Analysis. 2017
- Статический анализ // Wikipedia URL: https://ru.wikipedia.org/wiki/Статический_анализ_кода.
- Тонкости анализа исходного кода C/C++ с помощью cppcheck // https://habrahabr.ru/post/210256/

Спасибо за внимание

Александр Покид
Инженер-программист

+7 812 346-89-56
a.pokid@kpda.ru

www.kpda.ru
www.swd.ru

СВД Встраиваемые Системы

SWD Software