Для цитирования: Аксенов А.С. Анализатор структур данных прикладного уровня взаимодействия на основе автоматизированного извлечения знаний из исходных текстов // Вопросы радиоэлектроники. 2018. № 12. С. 45–49.
DOI 10.21778/2218-5453-2018-12-45-49
УДК 004.457

**А.С. Аксенов**[1]

[1] АО «Научно-производственное предприятие «Рубин»

# АНАЛИЗАТОР СТРУКТУР ДАННЫХ ПРИКЛАДНОГО УРОВНЯ ВЗАИМОДЕЙСТВИЯ НА ОСНОВЕ АВТОМАТИЗИРОВАННОГО ИЗВЛЕЧЕНИЯ ЗНАНИЙ ИЗ ИСХОДНЫХ ТЕКСТОВ

*Рассматривается вопрос создания программного средства, обеспечивающего проведение анализа данных, полученных по различным интерфейсам передачи информации. Проведение анализа позволяет выполнить проверку правильности формирования структуры информационных и управляющих сообщений различных компонентов разрабатываемой системы, а также отладку сетевого взаимодействия и наладку программных и аппаратных средств в части их информационного взаимодействия на уровне информационной совместимости. Выполнено сравнение с существующими средствами анализа сетевой активности и сопоставлено несколько подходов к решению вопроса анализа данных на прикладном уровне. Приведено обоснование выбора унифицированного формата представления исходных данных srcML. Указано направление дальнейшего развития программы анализатора в рамках представленного проектного решения. Отмечена целесообразность разработки данного программного средства и приведены результаты его применения при разработке программно-технических комплексов специального назначения.*

***Ключевые слова:*** *srcML, анализ данных, программное средство, расширенная грамматика*

**Введение**

При отладке сетевого взаимодействия часто возникает задача проверки правильности принимаемых/отсылаемых данных – порядка байтов, а также их значения. В общем случае это можно сделать с помощью различных анализаторов трафика (tcpdump, Wireshark и др.). Наиболее популярное графическое приложение для анализа трафика – Wireshark, которое понимает структуру самых различных сетевых (а с недавнего времени не только сетевых) протоколов и позволяет разобрать сетевой пакет, отображая значение каждого поля протокола любого уровня. Но когда дело доходит до седьмого уровня модели OSI [1] (уровня приложений), программа не может разобрать передаваемые данные на поля, так как не знает, согласно какому протоколу эти данные записаны (рис. 1).

Базовый функционал Wireshark может быть расширен с помощью плагинов, написанных на языке C или скриптовых языках Python и Lua. Однако такие плагины позволяют разобрать только заголовок проприетарного протокола без возможности идентифицировать поля полезной нагрузки. Разработка же плагина-диссектора для каждого

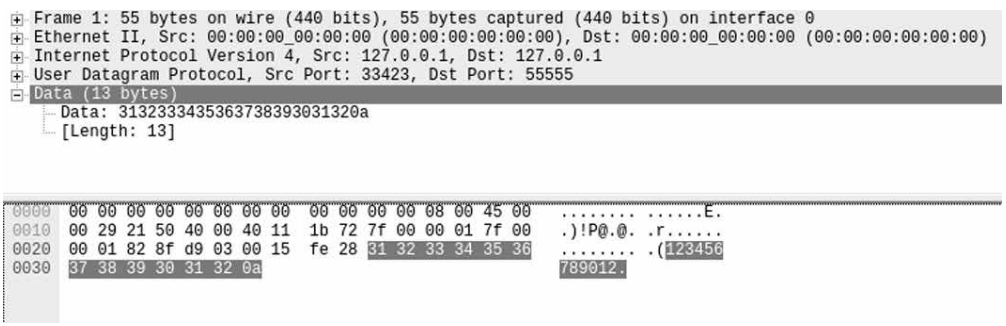


Рисунок 1. Анализ сообщения с помощью Wireshark

сообщения – трудоемкая задача при большом количестве сообщений. Трудозатраты будут неэффективными, если такой плагин будет иметь однократное применение.

Таким образом, актуальной задачей является разработка приложения для проверки правильности формирования данных в соответствии с используемым протоколом, которое обеспечивало бы удобное определение значений полей внутри полезной нагрузки, а также выполнения анализа данных, полученных через различные интерфейсы (Ethernet, RS-232/485 и т.д.).

**Новый анализатор структур данных**

Разработанное приложение получило название Sourceshark (в названии прослеживаются аналогии с Wireshark). Разработка производилась на языке C++ с использованием фреймворка Qt версии 5.9.2 [2]. Для функционирования программы необходимо наличие в системе свободно распространяемых библиотек boost версии не ниже 1.55.0 и srcML версии не ниже 0.9.5. Sourceshark позволяет реализовать проверку правильности отсылаемых/принимаемых данных в удобном виде, провести отладку аппаратуры, протоколов, программ взаимодействия между различными компонентами системы.

Принцип работы приложения построен на технологии извлечения знаний из исходных текстов с использованием промежуточного представления кода. Использование формата промежуточного представления обусловлено его универсальностью (отображение всех синтаксических конструкций языка, содержащихся в стандарте) и расширяемостью (имеется возможность введения новых конструкций языка при переходе на новую версию). Текстовый формат промежуточного представления позволяет использовать большое количество готовых инструментов для обработки текста.

Как правило, при реализации протокола взаимодействия между приложениями все пересылаемые данные «упакованы» в последовательности полей, описываемые в композитном типе данных – структуре (struct), хранящейся в заголовочных файлах (речь идет только о разработке на языках C/C++). Поэтому нет необходимости описывать протокол взаимодействия еще раз, как это нужно при создании плагина к Wireshark, следует только передать готовый заголовочный файл приложению и подгрузить в него массив пересылаемых данных.

Для получения промежуточного представления исходного кода предполагается использование абстрактного синтаксического дерева (abstract syntax tree – AST), дополненного семантической информацией [3]. Схематично данный процесс изображен на рис. 2.

Среди свободных инструментов для получения AST можно выделить следующие продукты: ROSE Compiler, Mozilla Dehydra & Treehydra (развитие прекращено в 2010 г.), Clang [4]. Лучше других для указанной решения задачи подходит Clang и его библиотека libclang, поддерживаемая активным сообществом и используемая крупными компаниями (например, Google, Apple) во многих проектах для статического анализа и автозавершения кода. Но, ввиду того что для извлечения AST с помощью Clang требуется полная завершенность единицы трансляции, указание путей поиска заголовочных файлов, включаемых в каждую единицу трансляции, так как по сути выполняется компиляция исходных текстов до стадии семантического анализа, от данного варианта пришлось отказаться. Выбор был сделан в пользу библиотеки srcML (Source Markup Language) [5], которая переводит исходный код в формат srcML.

Формат srcML представляет собой документоориентированное XML-представление исходного кода, дополненное информацией из AST, где тэги разметки идентифицируют элементы абстрактного синтаксиса языка. Грамматика srcML подобна грамматике расширенной Бэкус-Науровой формы (РБНФ) [6], объединяя известный синтаксис XML и РБНФ, чтобы выразить содержание отдельных элементов в терминах правил.

Особенность формата srcML заключается в том, что он сохраняет весь текст исходного кода, например комментарии, форматирование (пробелы), директивы препроцессора, обеспечивает полный доступ к исходному коду на лексическом и документальном уровнях с эквивалентным прямым и обратным отображением между исходным кодом и srcML. Эти элементы идентифицируются для дальнейшей обработки средами разработки и инструментами понимания программ. Тэги комментариев, директив препроцессора, операторов и другого синтаксиса

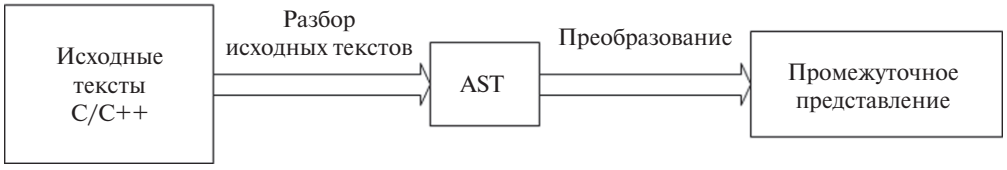


Рисунок 2. Алгоритм построения промежуточного представления исходного кода

позволяют получить доступ к исходному коду на документальном, структурном и синтаксическом уровнях через XML. Промежуточное представление и инструментарий устойчивы к нарушениям исходного кода: некомпилируемый код, фрагменты кода, одиночные объявления и одиночные файлы с представлением, основанным только на информации из локального документа, т. е. таблица символов, в которой в процессе работы компилятор хранит информацию об объектах, не используется.

Библиотека srcML предоставляет инструменты для эффективного преобразования файлов исходного кода в формат srcML со скоростью 25 KLOCs/сек (тысяч строк кода в секунду) и приблизительно 3000 файлов/мин. Например, весь исходный код ядра Linux преобразуется в формат srcML меньше чем за 7 мин. Также присутствует возможность обратной трансформации из формата srcML в исходный код со скоростью более 250 KLOCs/сек. Исходный код до обработки заголовочного файла показан на рис. 3.

```
#include <QVariant>

#define TEST_1 1

#pragma pack(push,1)

struct TestStructOne
{
   qint32 fieldInt32;
   qreal fieldReal;
   qint8 fieldInt8;
};

#pragma pack(pop)
```

Рисунок 3. Исходный код до обработки

После обработки исходный код принимает вид, отображенный на рис. 4. При таком унифицированном виде достаточно просто произвести анализ и фильтрацию по необходимым параметрам (тэгам). К такому же виду приводятся все включаемые заголовочные файлы, поэтому важно, чтобы все необходимые для правильного разбора заголовочные файлы лежали рядом с исходным файлом.

После открытия заголовочного файла программа Sourceshark производит поиск включений других заголовочных файлов, обеспечивает построение иерархичного списка всех зависимостей подключаемых файлов, после чего выполняет их поочередный рекурсивный анализ. Все анализируемые файлы передаются программой на обработку в библиотеку srcML. После получения промежуточного представления исходного кода программа обеспечивает анализ файла путем поиска всех структур, а также информации о наследовании других структур. При наличии информации о наследуемой структуре она заносится в описание анализируемой структуры. Таким образом, сохраняется иерархия данных. После завершения анализа имеется вся необходимая информация о структурах – имена и типы полей, входящих в состав (с учетом полей наследуемых структур), размер каждого поля, количество байтов выравнивания полей.

Как результат разбора всех файлов, в главном окне приложения Sourceshark в выпадающем списке будет находиться перечень всех найденных структур. Для визуального отображения полей структуры производится выбор необходимого сообщения в выпадающем списке. В секции «Поля структуры» отобразятся имя поля данных, его тип и цветовое обозначение. Если данные для анализа были подгружены ранее, то в секции «Данные» появится такое же цветовое обозначение полей структуры. Конечно, существуют некоторые ограничения, связанные с типом отображаемых полей, например, невозможно правильно отобразить размер строковых полей, так как он определяется размером передаваемых в них данных.

Для обеспечения приложения входными данными можно использовать программу Wireshark, в главном окне которой необходимо выбрать перехваченный пакет и извлечь данные из секции Data UDP-пакета в отдельный файл, выбрав в контекстном меню пункт Export Packet Bytes либо использовать уже подготовленный другими способами файл с данными.

Для проверки правильности переданных/принятых данных полученный файл загружается в приложение Sourceshark (рис. 5).

На данный момент реализована поддержка основных синтаксических конструкций языка стандарта C++03, в дальнейшем планируется развитие

```
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;QVariant&gt;</cpp:file></cpp:include>

<cpp:define>#<cpp:directive>define</cpp:directive> <cpp:macro><name>TEST_1</name></cpp:macro> <cpp:value>1</cpp:value></cpp:define>

<cpp:pragma>#<cpp:directive>pragma</cpp:directive> <name>pack</name><name>(</name><name>push</name><name>,</name><name>1</name><name>)</name></cpp:pragma>

<struct>struct <name>TestStructOne</name>
<block>{<public type="default">
   <decl_stmt><decl><type><name>qint32</name></type> <name>fieldInt32</name></decl>;</decl_stmt>
   <decl_stmt><decl><type><name>qreal</name></type> <name>fieldReal</name></decl>;</decl_stmt>
   <decl_stmt><decl><type><name>qint8</name></type> <name>fieldInt8</name></decl>;</decl_stmt>
</public>}</block>;</struct>

<cpp:pragma>#<cpp:directive>pragma</cpp:directive> <name>pack</name><name>(</name><name>pop</name><name>)</name></cpp:pragma>
```

Рисунок 4. Исходный код после обработки библиотекой srcML

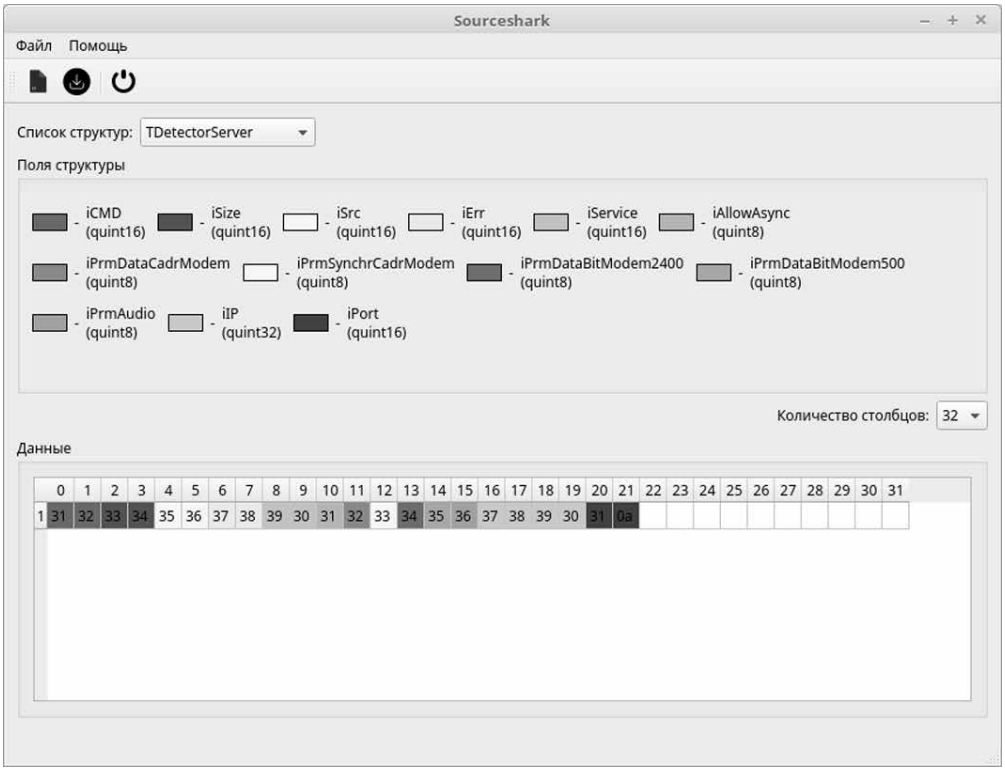


Рисунок 5. Анализ данных с помощью приложения Sourceshark

проекта для поддержки новых стандартов языка (C++11/14/17).

**Выводы**

Новое приложение успешно применяется при разработке программно-технических комплексов специального назначения. Данное решение предлагает удобный интерфейс для детального разбора данных, позволяет снизить трудозатраты при выполнении работ по выявлению ошибок в передаваемых данных, уменьшить время отладки сетевого взаимодействия между различными компонентами разрабатываемой комплексной аппаратной связи и в значительной степени оптимизировать процесс тестирования и наладки вновь разрабатываемых перспективных средств связи.

## СПИСОК ЛИТЕРАТУРЫ

1. Таненбаум Э., Уэзеролл Д. Компьютерные сети. СПб.: Питер, 2012. 960 с.
2. Qt | Crossplatform software development for embedded & desktop [Электронный ресурс]. URL: https://doc.qt.io/qt-5.9 (дата обращения: 10.10.2017).
3. Ахо А., Лам М., Сети Р., Ульман Д. Компиляторы. Принципы, технологии и инструментарий. М.: Вильямс, 2016. 1184 с.
4. Пустыгин А. Н., Ошнуров Н. А., Ковалевский А. А. Проект технологии извлечения знаний из исходных текстов на языках C++ и C# с использованием общего промежуточного представления // Матер. IX конф. «Свободное программное обеспечение в высшей школе» OSEDUCONF-2014 [Электронный ресурс]. URL: http://0x1.tv/20140126–2 (дата обращения: 15.08.2017).
5. srcML – Source Markup Language [Электронный ресурс]. URL: http://srcml.org (дата обращения: 20.08.2017).
6. ISO/IEC14977:1996(E) Information technology – Syntactic metalanguage – Extended BNF [Электронный ресурс]. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip (дата обращения: 23.08.2017).

## ИНФОРМАЦИЯ ОБ АВТОРЕ

**Аксенов Александр Сергеевич,** инженер-программист, АО «Научно-производственное предприятие «Рубин», Российская Федерация, 440000, Пенза, ул. Байдукова, д. 2, тел.: 8 (8412) 20-48-33, e-mail: aksenovpnz@gmail.com.

*For citation: Aksenov A.S. Analyzer of data structures at application level of interaction based on automated extraction of knowledge (information) from sources. Voprosy radioelektroniki, 2018, no. 12, pp. 45–49.*
DOI 10.21778/2218-5453-2018-12-45-49

**A.S. Aksenov**

## ANALYZER OF DATA STRUCTURES AT APPLICATION LEVEL OF INTERACTION BASED ON AUTOMATED EXTRACTION OF KNOWLEDGE (INFORMATION) FROM SOURCES

This paper considers the issue of creating a software tool that provides the performing of the analysis of data received via various data transfer interfaces. The performing of the analysis helps to check the correctness of formation of the structure of informational and controlling messages of various components of a system under development, as well as the correctness of the network interaction and testing debugging and adjustment of software and hardware in terms of their information interaction at the level of information compatibility. A comparison with the existing network activity analysis tools is presented and several approaches to solving the issue of data analysis at the application level are compared. The article provides the validity of the choice of a unified format of srcML source data representation. Also it specifies the directions for further development of the analyzing program within present project solution. The expediency of the development of this software tool and the results of its application in the development of special-purpose hardware and software suite are given in the conclusion.

**Keywords:** srcML, data analysis, software tools, advanced grammar

## REFERENCES

1. Tanenbaum A.S., Wetherall D.J. *Computer networks*. 5th ed. Pearson, 2012, 960 p.
2. Qt | Crossplatform software development for embedded & desktop. Available at: https://doc.qt.io/qt-5.9 (accessed 10.10.2017).
3. Aho A.V., Lam M.S., Sethi R., Ullman J.D. *Compilers: principles, techniques, and tools*. 2nd ed. Addison Wesley, 2006, 1040 p.
4. Pustygin A.N., Oshnurov N.A., Kovalevskii A.A. The project technology to extract knowledge from source code in C++ and C# using a common intermediate representation. (Conference proceedigs) IX konf. «Svobodnoe programmnoe obespechenie v vysshei shkole» OSEDUCONF-2014. (In Russian). Available at: http://0x1.tv/20140126–2 (accessed 15.08.2017).
5. srcML – Source Markup Language. Available at: http://srcml.org (accessed 20.08.2017).
6. ISO/IEC14977:1996(E) Information technology – Syntactic metalanguage – Extended BNF. Available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip (accessed 23.08.2017).

## AUTHOR

**Aksenov Alexander,** software engineer, JSC Research and Production Enterprise «Rubin», 2, Baydukova St, Penza, 440000, Russian Federation, tel.: +7 (8412) 20-48-33, e-mail: aksenovpnz@gmail.com.