Интернет-журнал «Науковедение» ISSN 2223-5167 http://naukovedenie.ru/

Том 8, №3 (2016) http://naukovedenie.ru/index.php?p=vol8-3

URL статьи: http://naukovedenie.ru/PDF/128TVN316.pdf

Статья опубликована 08.07.2016.

Ссылка для цитирования этой статьи:
Федотова Е.Л., Федотов А.А., Мадумарова К.В. Методика унификации процессов интерпретации программного кода // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 8, №3 (2016) http://naukovedenie.ru/PDF/128TVN316.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

УДК 004.415.2

Федотова Елена Леонидовна
ФГАОУ ВО «Национальный исследовательский университет «Московский институт электронной техники», Россия, Москва[1]
Кандидат педагогических наук, доцент
E-mail: fedotova-e2007@yandex.ru

Федотов Андрей Александрович
ФГАОУ ВО «Национальный исследовательский университет «Московский институт электронной техники», Россия, Москва
Кандидат технических наук
E-mail: andrey_fedotov_@mail.ru

Мадумарова Ксения Вениаминовна
ФГАОУ ВО «Национальный исследовательский университет «Московский институт электронной техники», Россия, Москва
Магистр
E-mail: kv.zemskova@mail.ru

Методика унификации процессов интерпретации программного кода

Аннотация. Универсальным средством для отладки и тестирования программного и аппаратного обеспечения является языковой процессор – программа или техническое средство, выполняющее программный код. Исключение необходимости обучаться новым языкам – огромное преимущество с точки зрения конкурентоспособности и доказывает актуальность данной работы. Для научного исследования используется многофакторный анализ. Проблема унификации процессов интерпретации программного кода состоит в том, что создание универсальных типов данных и способов их передачи потребует определённых издержек, таких как оперативная память и время интерпретации программного кода. Разработка данной методики позволяет операторам легче адаптироваться к работе с интерпретатором. Проблема связывания программного кода разных языков программирования имеет явно выраженный комплексный характер. Практическая ценность работы заключается в расширении возможностей универсальной интерпретации программного кода и повышении производительности работы программистов за счёт использования разработанного программного модуля в разных областях программирования, таких как модульное программирование, сопряжённое проектирование, тестирование, отладка.

[1] 124365, Москва, Зеленоград, корпус 2010, кв. 38

Ключевые слова: процессор; программный код; программное обеспечение; унификация; многофакторный анализ; модульное программирование; интеграция модулей; универсальный командный интерпретатор; сопряженное проектирование; универсальная интерпретация программного кода

В настоящее время широкое распространение получило модульное программирование. В результате аналитического исследования концепцию модульного программирования можно сформулировать в виде нескольких понятий и положений.

1. Функциональная декомпозиция задачи - разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач - модулей. Модули связаны между собой только по входным и выходным данным.
2. Модуль – основа концепции. Каждый модуль в функциональной декомпозиции представляет собой “чёрный ящик” с одним входом и одним выходом. Модульный подход позволяет производить модернизацию программы в процессе ее эксплуатации и облегчает ее сопровождение.

Два основных аспекта модульного программирования на разных языках – это формализм, доступный для обозначения и использования объектов других языков, а также инфраструктура для работы с такими объектами. Взаимодействие может охватывать различные аспекты, такие как доступ к сторонним данным, представление сторонних типов, вызовы сторонних подпрограмм, обработка сторонних событий, таких как исключения, и повторное использование иерархий классов объектов. В любом случае, взаимодействие всегда подразумевает соглашение между заинтересованными сторонами. Например, вызов подпрограммы будет работать правильно только тогда, когда вызывающий и вызываемый договорились о том, как передаются аргументы (в каком порядке, используя какие вычислительные ресурсы), кто размещает, удаляет ту или иную часть стека, как работает указатель стека, и т.д.

Проблема интеграции модулей и частей систем решается разными способами. В распределенных системах вводят промежуточный уровень программного обеспечения, который в сетевых операционных системах позволяет более или менее скрыть от пользователя разнородность набора аппаратных платформ и повысить прозрачность распределения. Многие современные распределенные системы построены в расчёте на этот дополнительный уровень, который получил название программного обеспечения промежуточного уровня [2]. Объектно-ориентированная технология показала своё значение при разработке нераспределенных приложений. Одним из наиболее важных свойств объекта является то, что он скрывает своё внутреннее строение от внешнего мира посредством строго определённого интерфейса. Такой подход позволяет легко заменять или изменять объекты, оставляя интерфейс неизменным. Ключевая особенность объекта состоит в том, что он инкапсулирует данные, называемые состоянием, и операции над этими данными, называемые методами. Доступ к методам можно получить через интерфейс. Объект может реализовывать множество интерфейсов. Точно так же для данного описания интерфейса может существовать несколько объектов, предоставляющих его реализацию. Это подразделение на интерфейсы и объекты, реализующие их, очень важно для распределенных систем.

В настоящее время на предприятиях, занимающихся разработкой и производством устройств и блоков вычислительной аппаратуры, используется сопряженное проектирование. Универсальным средством для отладки и тестирования программного и аппаратного обеспечения является языковой процессор – программа или техническое средство, выполняющее программный код. Сегодня существует множество языков программирования,

пригодных для отладки аппаратуры. Исключение необходимости обучаться новым языкам – огромное преимущество с точки зрения конкурентоспособности.

Все вышесказанное позволяет сделать вывод об актуальности создания теоретических основ и прикладных методов унификации процессов интерпретации программного кода.

Объектом исследования являются средства интерпретации программного кода.

Предметом исследования являются методы и средства унификации процессов интерпретации командного кода, а также оценка возможных потерь в производительности, как следствие унификации этих процессов. Решение этой проблемы позволит пользователю легче адаптироваться к работе с интерпретатором, так как пропадет необходимость в обучении новым языкам программирования.

Очевидно, с увеличением сложности программы, производительность неизбежно упадет. Важно оценить какое именно влияние окажет унификация доступа на скорость интерпретации программного кода.

Целью данной работы является повышение производительности работы пользователя как следствие унификации процессов интерпретации, а также исследование влияния унификации доступа на быстродействие программного модуля. Необходимо разработать систему универсального хранения данных и функций; сравнить быстродействие отдельных интерпретаторов с быстродействием универсального командного интерпретатора, оценить преимущества и недостатки выбранного подхода.

В соответствии с целью и предметом исследования возникает необходимость решить следующие задачи:

- исследование основных характеристик отдельных программных интерпретаторов;
- анализ производительности существующих программных интерпретаторов для выбранных языков программирования;
- разработка внешнего интерфейса(front-end), а также внутренних интерфейсов (back-end) для выбранных языков программирования в программном модуле «Универсальный командный интерпретатор»;
- сравнение быстродействия программ с унифицированным доступом и отдельно взятых интерпретаторов.

Для научного исследования используется многофакторный анализ программного модуля [3]. Для сравнения его быстродействия с быстродействием стандартных интерпретаторов будут использованы замеры времени, потраченного на одинаковые тексты программ. После первого этапа измерений, программный модуль будет оптимизирован, а эксперимент – повторен.

Многие современные компьютерные приложения используют модули, написанные на разных языках программирования. И интеграция этих модулей – деликатная операция. В первую очередь требуется наличие договоренностей, чтобы дать возможность программистам обозначить «сторонние» сущности как объекты или функции, а также связанные с ними типы данных. Необходимо обеспечить возможность создания иерархических структур, и правильного передачи данных: ссылок, указателей, если потребуется. Также необходимо перевести то, чем будут часто пользоваться программисты на уровень интерфейсов, вплоть до спецификации ABI (Application Binary Interfaces, Двоичный интерфейс приложения).

Переход от классического к сопряженному проектированию требует серьезного пересмотра методов разработки. Моделирование системной архитектуры с учетом всех параметров модели и последующие изменения этой архитектуры могут занять значительное количество времени по сравнению с разработкой архитектуры в классическом проектировании. Функциональные технические требования к исполнению могут использоваться как эталонная модель в процессе разработки программного обеспечения, включая программируемое оборудование и тестовую среду. Одновременное моделирование аппаратной и программной частей позволяет на ранних стадиях обнаружить ошибки в тестах или циклах разработки.

Проблема унификации процессов интерпретации программного кода состоит в том, что создание универсальных типов данных и способов их передачи потребует определённых издержек, таких как оперативная память и время интерпретации программного кода. Разработка такой методики позволит операторам легче адаптироваться к работе с интерпретатором, так как пропадёт необходимость в обучении новым языкам программирования. Важно оценить, какое именно влияние окажет унификация доступа на скорость интерпретации программного кода.

Процесс связывания программного кода разных языков программирования имеет явно выраженный комплексный характер и включает ряд основных этапов:

1) разделение обязанностей языковых процессоров на внутренние и внешние;
2) создание механизмов работы с данными: общих переменных, функций, создание механизмов передачи данных: контекст, стек значений, создание объектов, контролирующих процесс разбора кода.

В данной работе проведен анализ существующих инструментов, использующих или описывающих унификацию данных и процессов разбора программного кода [4]. В результате проведенного анализа были получены следующие результаты:

1. Рассмотрены существующие средства связывания данных: технология CORBA, инструмент SWIG, язык QtScript, рассмотрены способы хранения абстрактных данных в фреймворке Qt. Рассмотрены объекты типа QVariant, их создание, конвертирование, копирование. Возможность хранения в контейнерах других контейнеров.
2. Рассмотрены основные принципы разделения процесса разбора кода на отдельные этапы, а также особенности каждого этапа.

Универсальный доступ к процессам интерпретации может быть обеспечен путем выделения общих частей программного модуля в отдельный блок – внешний интерфейс – front-end. При разработке программного модуля «Универсальный командный интерпретатор», таким блоком стала библиотека SmMultiScriptFrontEnd.

Внешний интерфейс обеспечивает всю необходимую работу по подключению внутренних интерфейсов языков, определению синтаксиса конкретного файла, предоставляет средства для запуска процессов вычисления и выполнения программных текстов, а также предоставляет систему оповещения об ошибках и сообщениях. Важной частью внешнего интерфейса является универсальное представление данных. Переменные, функции и другие элементы языка программирования должны быть представлены максимально унифицировано и при этом сохранять свою структуру и особенности работы.

Рассмотрены средства универсальной интерпретации [5].

1. Мультиязыковое программирование – формализм, доступный для обозначения и использования “сторонних” объектов других языков, а также инфраструктура для работы с такими объектами. Этот термин вводят Cyrille Comar, Matthew Gingell, Olivier Hainque, Javier Miranda в статье “Multi-Language Programming: The Challenge and Promise of Class-Level Interfacing”[6].

2. Фреймворк Qt - кросплатформенный инструментария разработки ПО на языке программирования C++. Этот фреймворк обладает встроенным интерпретатором JavaScript, который позволяет интегрировать код на данном языке в нативный код на языке C++.

Два основных аспекта мультиязыкового программирования – это формализм, доступный для обозначения и использования “сторонних” объектов других языков, а также инфраструктура для работы с такими объектами. Взаимодействие может охватывать различные аспекты, такие как доступ к сторонним данным, представление сторонних типов, вызовы сторонних подпрограмм, обработка сторонних событий, таких как исключения, и повторное использование иерархий классов объектов. В любом случае, взаимодействие всегда подразумевает соглашение между заинтересованными сторонами. Например, вызов подпрограммы будет работать правильно только тогда, когда вызывающий и вызываемый договорились о том, как передаются аргументы (в каком порядке, используя какие вычислительные ресурсы), кто размещает, удаляет ту или иную часть стека, как работает указатель стека, и т.д.

Первый набор основных возможностей взаимодействия обеспечивается явными особенностями языка программирования, связанными с хорошо известными соглашениями о вызовах для целевой среды, указанной в базовых документах ABI. В большинстве соглашений находятся различные стандартные способы, доступные на стороне языка программирования. В качестве первого примера, Справочное руководство Ada включает в себя полное приложение, посвященное этому вопросу, охватывающее взаимодействие с C, Cobol, Fortran и позволяющее реализовать поддержку других языков.

Минимальная поддержка, указанная в этом приложении, состоит из стандартных пакетов для каждого языка, например в Interfaces.C иерархия для C, и специфичные директивы компилятора:

- директива Import, чтобы загрузить объект, определенный на стороннем языке в программу на Ada, таким образом, позволяя подпрограммам, написанным на стороннем языке, быть вызванным из Ada;
- директива Export, чтобы экспортировать объект Ada стороннему языку;
- директива Convention, чтобы указать, что объект Ada должен использовать соглашения другого языка для передачи параметров в подпрограммы, либо представлять тип данных в памяти (например, определяющий порядок элементов матрицы);
- директива Linker Options, чтобы задать параметры системы, необходимые при компоновке программы.

В процесс включается для вызова функции, написанной на C, из Ada, чтобы распечатать числовое значение, находящееся в указанном адресе. Там используется стандартный пакет Interfaces. C для получения доступа к типу в Ada, соответствующему типу int, объявляет подпрограмму Ada для перевода сервисов C, и импортирует сервис с помощью директивы Import. Последняя говорит компилятору, что подпрограмма является внешней с соглашением C и устанавливает, какой символ (имя связи) должен быть использован, чтобы обратиться к ней with Interfaces.C; use__.

Проблема связывания программного кода разных языков программирования явно выраженный комплексный характер и включает ряд основных этапов:

- разделение обязанностей языковых процессоров на внутренние и внешние;
- создание механизмов работы с данными: общих переменных, функций, создание механизмов передачи данных: контекст, стек значений, создание объектов, контролирующих процесс разбора кода.

В работе проведён анализ существующих инструментов, использующих или описывающих унификацию данных и процессов разбора программного кода. В результате проведённого анализа были получены следующие результаты [6]:

1. Рассмотрены существующие средства связывания данных: технология CORBA, инструмент SWIG, язык QtScript, рассмотрены способы хранения абстрактных данных в фреймворке Qt. Рассмотрены объекты типа QVariant, их создание, конвертирование, копирование. Возможность хранения в контейнерах других контейнеров.
2. Рассмотрены основные принципы разделения процесса разбора кода на отдельные этапы, а также особенности каждого этапа.

На основе проведенного анализа сформулированы цели и задачи исследования, состоящие в улучшении характеристик универсального интерпретатора, повышения производительности его пользователей за счет удобства использования нескольких языков программирования, разработке программных компонентов для языков SmScript, JavaScript, а также в исследовании влияния унификации доступа на быстродействие программного модуля.

Универсальный командный интерпретатор представляет собой динамическую библиотеку, подключаемую к любой другой программе. Эта библиотека предназначена для предоставления единого интерфейса передачи данных и управления во внутренние интерфейсы разных языков программирования.

Использование универсального командного интерпретатора предоставляет следующие возможности: − загрузка динамических библиотек для языков программирования; − передача данных в универсальном виде для использования их как в основном модуле программы, так и во внутренних интерфейсах; − передача функций в универсальном виде для использования их в программном коде, написанном на разных языках программирования; − запуск процесса трансляции программного кода; − прерывание процесса трансляции программного кода; − запуск процесса исполнения программного кода; − прерывание процесса исполнения программного кода.

В процессе разработки и исследований получены следующие научные результаты:

1. Поставлена проблема универсальной интерпретации программного кода.
2. Предложена новая методика для универсальной интерпретации программного кода.
3. Разработан программный модуль «Универсальный командный интерпретатор» (ПМ УКИ).
4. Программный модуль расширен для поддержки нескольких новых языков программирования.
5. Проведено тестирование производительности программного модуля для разных языков программирования на задачах, имеющих разную вычислительную сложность.

Практическая ценность работы заключается в расширении возможностей универсальной интерпретации программного кода и повышении производительности работы программистов за счёт использования разработанного программного модуля в разных областях программирования, таких как модульное программирование, сопряжённое проектирование, тестирование, отладка.

ЛИТЕРАТУРА

1. Альфред В. Ахо. Компиляторы: принципы, технологии и инструментарий, 2-е изд.: Пер. с англ. / Ахо, Альфред В., Лам, Моника С., Сети, Рави, Ульман, Джеффри Д. – М.: ООО “И.Д. Вильямс” 2015. - 1184 с.
2. Гагарина Л.Г., Киселев Д.В., Федотова Е.Л. Разработка и эксплуатация автоматизированных информационных систем: учеб. пособие / под ред. проф. Л.Г. Гагариной М.: ИД ФОРУМ: ИНФРА-М, 2014, 384 с.
3. Э. Гамма. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес – СПб., 2010.
4. Истомин Е.П. Высокоуровневые методы информатики и программирования: Учебник / Е.П. Истомин, В.В. Новиков, М.В. Новикова - СПб. ООО «Андреевский издательский дом» 2006 г. - 228 с.
5. Э. Таненбаум. Распределенные системы: принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – СПб.: Питер, 2003. - 877 с.
6. Дж. Хопкрофт Введение в теорию автоматов, языков и вычислений, 2-е изд. / Хопкрофт, Джон, Э., Мотвани, Раджив, Ульман, Джеффри, Д. - М. Издательский дом «Вильямс», 2002. - 528 с.
7. Федотова Е.Л. Информационные технологии и системы: учеб. пособие. - М.: ИД ФОРУМ: ИНФРА-М, 2009, 352 с.
8. Федотова Е.Л., Федотов А.А. Информатика. Курс лекций / Е.Л. Федотова, А.А. Федотов учеб. пособие. - М.: ИД ФОРУМ: ИНФРА-М, 2011, 480 с.
9. Федотова Е.Л., Портнов Е.М. Прикладные информационные технологии. учеб. пособие. - М.: ИД ФОРУМ: ИНФРА-М, 2013, 365 с.
10. Jurgen Teich. Hardware / Software Codesign: The Past, the Present, and Predicting the Future // Proceedings of the IEEE. – 2012. – Vol. 100.

Fedotova Elena Leonidovna
National research university of electronic technology, Russia, Moscow
E-mail: fedotova-e2007@yandex.ru

Fedotov Andrey Alexsandrovich
National research university of electronic technology, Russia, Moscow
E-mail: andrey_fedotov_@mail.ru

Madumarova Ksenua Veniaminovna
National research university of electronic technology, Russia, Moscow
E-mail: kv.zemskova@mail.ru

Methods of unification of processes of interpretation of the code

Abstract. A versatile tool for debugging and testing of hardware and software is a language processor program or a technical device that executes program code. Eliminate the need to learn new languages is a huge advantage in terms of competitiveness and proves the relevance of this work. For research used multivariate analysis. The problem of unification of processes of interpretation of software code is that the creation of a universal data types and methods of transfer will require certain costs such as memory and time interpretation of code. The development of this technique will allow operators to better adapt to the shell. Problem linking program code in various programming languages has a pronounced complex character. Practical value of work consists in extending the universal interpretation of program code and increase productivity of programmers through the use of the developed software in different areas of programming, such as modular programming, designing, testing, debugging.

Keywords: processor; program code; software; standardization; multivariate analysis; modular programming; integration of modules; the universal shell; paired design; the universal interpretation of the code

REFERENCES

1. Al'fred V. Akho. Kompilyatory: printsipy, tekhnologii i instrumentariy, 2-e izd.: Per. s angl. / Akho, Al'fred V., Lam, Monika S., Seti, Ravi, Ul'man, Dzheffri D. – M.: OOO "I.D. Vil'yams" 2015. - 1184 s.
2. Gagarina L.G., Kiselev D.V., Fedotova E.L. Razrabotka i ekspluatatsiya avtomatizirovannykh informatsionnykh sistem: ucheb. posobie / pod red. prof. L.G. Gagarinoy M.: ID FORUM: INFRA-M, 2014, 384 s.
3. E. Gamma. Priemy ob"ektno-orientirovannogo proektirovaniya. Patterny proektirovaniya / E. Gamma, R. Khelm, R. Dzhonson, D. Vlissides – SPb., 2010.
4. Istomin E.P. Vysokourovnevye metody informatiki i programmirovaniya: Uchebnik / E.P. Istomin, V.V. Novikov, M.V. Novikova - SPb. OOO «Andreevskiy izdatel'skiy dom» 2006 g. - 228 s.
5. E. Tanenbaum. Raspredelennye sistemy: printsipy i paradigmy / E. Tanenbaum, M. van Steen. – SPb.: Piter, 2003. - 877 s.
6. Dzh. Khopkroft Vvedenie v teoriyu avtomatov, yazykov i vychisleniy, 2-e izd. / Khopkroft, Dzhon, E., Motvani, Radzhiv, Ul'man, Dzheffri, D. - M. Izdatel'skiy dom «Vil'yams», 2002. - 528 s.
7. Fedotova E.L. Informatsionnye tekhnologii i sistemy: ucheb. posobie. - M.: ID FORUM: INFRA-M, 2009, 352 s.
8. Fedotova E.L., Fedotov A.A. Informatika. Kurs lektsiy / E.L. Fedotova, A.A. Fedotov ucheb. posobie. - M.: ID FORUM: INFRA-M, 2011, 480 s.
9. Fedotova E.L., Portnov E.M. Prikladnye informatsionnye tekhnologii. ucheb. posobie. - M.: ID FORUM: INFRA-M, 2013, 365 s.
10. Jurgen Teich. Hardware / Software Codesign: The Past, the Present, and Predicting the Future // Proceedings of the IEEE. – 2012. – Vol. 100.