УДК 004.02

ОБЗОР СПОСОБОВ ПОДДЕРЖКИ ПРОИЗВОЛЬНЫХ ТИПОВ В СКРИПТОВЫХ ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

Баранов В.Г. (ИТМО)
Кореньков Ю. Д. (ИТМО)
Научный руководитель – к.т.н., доцент, Кореньков Ю. Д. (ИТМО)

Современные скриптовые языки широко применяются для автоматизации задач, обработки данных и управления различными системами. Одной из важных возможностей таких языков является поддержка пользовательских типов данных, позволяющая формировать структуры данных и объектные модели. Однако, скриптовые системы, предназначенные для поддержки и администрирования ОС, такие как shell-скрипты и batch-файлы, не поддерживают пользовательские типы данных, что существенно ограничивает их возможности, и препятствует использованию нетривиальных скриптов в минималистичных окружениях, таких как контейнеры и бездисковые системы. Отсутствие статической проверки даже для поддерживаемых типов также усложняет использование таких скриптовых систем.

В данной работе рассматривается способ добавления поддержки пользовательских типов в скриптовый язык Amber со статической типизацией. Скрипты, написанные на данном языке могут выполняться двумя способами: интерпретатором или посредством трансляции в файл скрипта на языке Bash. Данный подход аналогичен тому, как файл с кодом на скриптовом языке TypeScript может быть выполнен с помощью ts-node или транслирован выходной файл скрипта на языке JavaScript. Статическая типизация в языке Amber позволяет исключить некоторые виды ошибок в скриптах на этом языке так же, как элементы статической типизации в языке TypeScript исключают некоторые виды ошибок в скриптах на этом языке по сравнению с целевым JavaScript. При этом такой подход позволяет значительно расширить возможности Bash-скриптов, сохраняя их совместимость с существующей инфраструктурой.

Поддержка пользовательских типов данных в различных популярных скриптовых языках с достигается схожим образом:

- В Python используются встроенные структуры данных на основе dict (хэш-таблица) и list (связный список).
- В JavaScript пользовательские типы данных реализуются за счёт объектов, представленных как ассоциативные массивы.
- Lua использует таблицы (table) как универсальная структура данных, позволяющая моделировать объекты, массивы и словари.
- В языке Ruby поддержка пользовательских типов данных также, как и в Python, достигается засчет хэш-таблиц.

Язык Amber, компилируемый в Bash, тажке поддерживает статическую типизацию, однако у него нет поддержки пользовательских типов данных. При этом, в отличии TypeScript и JavaScript, использующийся для выполнения скриптов на Amber, язык shell-скриптов Bash поддержкой пользовательских типов данных не обладает.

Предлагаемое решение состоит из двух этапов. Первый этап – добавление в систему типов и синтаксическую модель языка Amber необходимых синтаксических конструкций и проверок, дающих возможность его пользователю формировать классы объектов, обладающих методами и полями, и контролировать правильность их использования. Второй этап – организация поддержки пользовательских типов данных во время выполнения на основе shell-скриптов, распространённые интерпретаторы которых таковой на данный момент не обладают.

Язык shell-скриптов поддерживает простые подпрограммы и переменные, с которыми могут ассоциироваться текстовые значения. Кроме этого, интерпретаторы shell-скриптов поддерживают установку и чтение значения переменной, имя которой формирует динамически посредством конкатенации строк. Используя эту базовую функциональность, можно написать набор подпрограмм для чтения и установки значения полей, принимающих

аргументами идентификатор объекта и имя поля, и использующие существующие операции чтения и установки переменных, имена которых формируются путём конкатенации предопределённого префикса, идентификатора объекта и имени поля, принадлежащего данному объекту. Дополнительно можно реализовать набор подпрограмм, осуществляющих порождение и разрушение такого объекта: установку набора соответствующих состоянию объекта переменных в начальные значения и их полный сброс.

Таким образом, за счёт небольшого набора специальных подпрограмм возможно разделить пространство глобального словаря переменных в интерпретаторе shell-скрипта на логические «маленькие» словари, соответствующие отдельным пользовательским объектам. Используя этот подход в качестве основы, возможно организовать не только поддержку времени выполнения для пользовательских типов данных исходного транслируемого в shell-скрипт языка, но и обеспечить возможность динамической проверки таких пользовательских типов, реализовав различные виды приведения и преобразования типов данных во время выполнения скриптов.

Рассмотренный подход также может является основой для создания трансляторов других языков с поддержкой пользовательских типов данных в shell-скрипты, так как поддержка пользовательских типов данных играет одну из ключевых ролей при их использовании. В Python, JavaScript, Lua и Ruby эта поддержка реализована через встроенные механизмы динамической типизации и ассоциативные структуры данных. Это даст возможность выполнять скрипты, написанные на различных языках в окружениях, где их интерпретаторы по каким-то причинам не доступны.

В результате работы была реализована динамическая поддержка пользовательских типов данных для языка shell-скриптов, использованная для добавления соответствующей в функциональности в язык Amber. Его использование позволяет увеличить удобство работы с данными и сократить время отладки скриптов с сохранением совместимости с POSIX-оболочкой, что облегчает выполнение задач, связанных с поддержкой и администрированием различных систем.

Список использованных источников:

1. Aho A., Lam M., Sethi R., Ullman J. Compilers: Principles, Techniques, and Tools. – Addison-Wesley, 2006
2. J. Cannon, Shell Scripting: How to Automate Command Line Tasks Using Bash Scripting and Shell Programming.
3. Cooper K., Torczon L. Engineering a Compiler. – Morgan Kaufmann, 2011.