UDC 004.438Lua

LUA MULTIPARADIGM PROGRAMMING LANGUAGE: APPLICATION AND OPPORTUNITIES

VITALY GAVRILOV, ARKADZI OSKIN
Polotsk State University, Belarus

Lua is one of the most popular embedded programming languages. This language is used in many different applications. It combines a small amount of memory, high speed, ease of use and great flexibility.

The effectiveness of a programmer's work depends largely on the correct choice of tools in relation to the problem being solved. The choice of programming language is a key position which the success of the implementation of a ready-made software system essentially depends on.

Today the total number of developed programming languages is estimated at several thousands. When classifying them, various selection criteria (applicability to the subject area, performance, scalability, etc.) are distinguished, grouping them together (according to the programming paradigms used, the method of execution, the principles of memory organization, etc.). Modern information technologies use only dozens of the most commonly used programming languages [1]. This article discusses one of the popular embedded programming languages – Lua [2].

Quite often, when developing a software product, you have to simplify the work of writing end user applications as much as possible. One of the possible solutions in such cases is the use of simple scripting programming languages, such as Lua. Script languages allow you to think less about specific data types, byte order and other internal features of the platform. Typically, this code is less cumbersome and easier to perceive.

This script language has a simplified syntax and provides language learning in the shortest possible time, it also has a small interpreter size, relatively high execution speed and easy extensibility. This language allows you to pay special attention to the algorithm and logic of work, and minimize the dependence of software on the architectural features of the platform used.

The authors of the language Luiz Henrique de Figueiredo, Waldemar Celes and Roberto Ierusalimschy from the very beginning – the year of its creation is 1993 – developed it for integration with software written in C / C++ and other common languages [3, 4]. This integration has many advantages. Lua is a small and simple programming language that complements C / C++ well. This is an interpreted language, which means that programs written in it are converted into bytecode directly during execution, without prior compilation. It has a safe environment, automatic memory management and good capabilities for working with strings and other resizable data types. In C / C++, a compiled language with static typing, various software components are created, and Lua is used to connect them. Lua, being a full-fledged programming language, gives a final form to the application, with the help of which adaptation and dynamic adjustment of these components is carried out, and completely new application components are created.

Despite the fact that the language is procedural and does not support the principles of object-oriented programming, some features of OOP (for example, inheritance) are fully performed by standard means of the language [5, P. 138].

Lua is an embedded scripting language, flexible and fast, easily portable, combining simple procedural syntax with powerful descriptions of data structures. Due to the fundamental concept of the language, Lua is a multiparadigm language and it can implement object-oriented and functional approaches to programming [6, P. 60].

One can extend the capabilities of Lua by writing their own libraries, which is one of the basic principles laid down during the language creation phase. Even the basic functionality is implemented as a set of standard libraries, for example, working with strings. There are also various third-party libraries. Lua was designed to be easily integrated into other applications. A well-designed interface allows the Lua code to fully interact with external code, so it integrates with a large number of languages, such as C / C++, Java, C#, Smalltalk, Fortran, Ada, Erlang, Perl and Ruby.

The Lua engine is written in pure C and all that is required to run Lua programs is the presence of a C compiler to build an interpreter. Due to this, high portability of the code to various platforms is achieved.

Lua is not the only scripting language. There are other languages that can be used for the same purpose. Nevertheless, Lua provides a whole range of features that make it the best choice for many tasks:

— *Extensibility*. Lua's extensibility is so remarkable that many people regard Lua not as a language, but as a kit for building domain-specific languages. Lua was designed from scratch to be extended, both through Lua code and through external C code. As a proof of the concept, Lua implements most of its own basic functionality through external libraries. It is really easy to interface Lua with C / C++, and Lua has been used integrated with several other languages as well, such as Fortran, Java, Ada, C#, and even with other scripting languages, such as Perl and Python.

— *Simplicity.* Lua is a simple and small language based on a small number of definitions. It has few (but powerful) concepts. This simplicity makes Lua easy to learn and contributes to its small size. The size of the Lua kernel version 5.2.3 along with the standard libraries is about 182 KB [6, P. 61].

— *Efficiency.* Lua has quite an efficient implementation. Independent benchmarks show that Lua is one of the fastest languages in the realm of scripting languages [7, P. 13].

— *Portability.* Starting Lua is possible on all platforms: different versions of UNIX and Windows, PlayStation, Xbox, Mac OS X and iOS, Android, Kindle Fire, NOOK, Haiku, QUALCOMM Brew, IBM mainframes, RISC OS, Symbian OS, Rabbit processors, Raspberry Pi, Arduino, and many others. The source code for each of these platforms is virtually the same. Lua does not use conditional compilation to adapt its code to different machines; instead, it sticks to the standard ANSI (ISO) C. It does not need to be adapted to the new environment, it can simply be compiled ANSI C. [7, P. 14]

The use of Lua usually belongs to one of three broad groups: a) those that use Lua already embedded in an application program, b) those that use Lua stand alone, c) those that use Lua and C together.

Many people use Lua embedded in any application, such as Adobe Lightroom, Nmap, Wireshark, VLC media player, and computer games, such as World of Warcraft, Angry Birds, Diablo 3, Far Cry and S.T.A.L.K.E.R. (in 2003, Lua was recognized as the most popular scripting language for game development according to the GameDev.net community) [8, P. 1159]. These applications use the Lua-C API to register new functions, to create new types, and to change the behaviour of some language operations, configuring Lua for their specific domains. Frequently, the users of such applications do not even know that Lua is an independent language adapted for a particular domain.

The capabilities of Lua as an independent language are not limited to text processing and small programs, but also apply to medium and large size projects. For such use there is a wide range of additional libraries. Lua Rocks, a system for building and managing modules for Lua, now has more than 150 packages.

Although Lua is an interpreted language, it always precompiles the source code into bytecode before it is executed. If necessary, the developer can also create bytecode using the luac compiler, and the interpreter can execute the file both as bytecode and as a source file. For time-critical tasks, there is a Lua JIT compiler – LuaJIT.

A large number of programmers use Lua as a library for C / C++. They write various applications in C / C++, and use Lua, a well-integrated programming language, to create simple and lightweight interfaces. An example of this is the AutoPlay Media Studio application designer from Indigo Rose Software. Application components are created in C / C++, and their adjustment, adaptation and dynamic changes are carried out using Lua [9].

REFERENCES

1. Абасова, Н.И. Анализ рейтингов языков программирования при их выборе / Н.И. Абасова, Н. А. Лаврухина // Информационные технологии и проблемы математического моделирования сложных систем. — 2009. — № 7. — С. 5—13. — [Electronic resource]. — Mode of access: https://elibrary.ru/download/elibrary_26151506_67327706.pdf. — Date of access: 20.01.2019.
2. The Programming Language Lua. — [Electronic resource]. — Mode of access: http://www.lua.org/. — Date of access: 02.12.2018.
3. Ierusalimschy, R., de Figueiredo, L. H., Celes, W. The Evolution of an Extension Language: A History of Lua / Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes. — 2001. — [Electronic resource]. — Mode of access: https://www.academia.edu/15734987/The_evolution_of_an_extension_language_A_history_of_Lua. — Date of access: 02.12.2018.
4. Ierusalimschy, R., de Figueiredo, L. H., Celes, W. The Evolution of Lua / Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes. — 2007. — [Electronic resource]. — Mode of access: https://www.academia.edu/15735031/The_evolution_of_Lua. — Date of access: 02.12.2018.
5. Миканович, А. Использование скриптового языка Embedded Lua во встраиваемых системах / Антон Миканович // Компоненты и технологии. — 2012. — № 11 (136). — С. 138—141.
6. Рачаев, К. Смотрим на «Луну» / К. Рачаев // Системный администратор. — 2014. — № 12 (145). – С. 60—65.
7. Иерузалимски, Р. Программирование на языке Lua / Р. Иерузалимски. — 3-е изд.— М: ДМК Пресс, 2016. — 382 с.
8. Ierusalimschy, R., de Figueiredo, L. H., Celes, W. The Implementation of Lua 5.0 / Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes // Journal of Universal Computer Science. — Vol. 11. — 2005. — No. 7. — P. 1159—1176.
9. Indigo Rose — Software Deployment Tools for Windows Developers. — [Electronic resource]. — Mode of access: https://www.indigorose.com/autoplay-media-studio/. — Date of access: 20.01.2019.