УДК 004

Подключение к Python проекту совместно используемой библиотеки

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье приведено написание компонентов на языке программирования C в виде файлов динамической библиотеки и интеграция полученных зависимостей в проект Python. Язык Python используется в качестве интерфейса и точки входа в программу. В результате исследования подробно описаны компиляция, загрузка и вызов функций библиотеки.
Ключевые слова: C, Python, динамическая библиотека.

Connecting a shared library to a Python project

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the writing of components in the C programming language in the form of dynamic library files and the integration of the resulting dependencies into a Python project. The Python language is used as the interface and entry point to the program. As a result of the study, the compilation, loading and calling of library functions will be described in detail.
Keywords: C, Python, shared library.

1 Введение

1.1 Актуальность

Компоненты динамической библиотеке написанные на языке C, приводят к большому увеличению скорости, по сравнению с реализацией используемых функций на языке Python. Так как Python является интерпретируемым языком программирования, то возникает сложность при работе с большим количеством вычислений. Функции, написанные на языках C/C++, компилируются в машинный код, что позволяет быстро вычислять длинные цепочки данных и встраивать в код в интерфейс других языков программирования. Другим не менее важным плюсом является безопасность скрытия исходного кода. Процесс реверс-инжиниринга будет очень сложен или даже практически невозможен, нежели при преобразовании исходного Python кода в исполняемый файл.

1.2Обзор исследований

С.И.Попков написал программную реализацию межъязыкового взаимодействия на базе динамических библиотек [1]. В.Ю.Ильичев, Ю.М.Жукова, И.В.Шамов исследовали вейвлет-анализ непериодических сигналов с применением совместно используемых библиотек [2]. М.Г.Хмельнюк, Д.И.Важинский использовали библиотеки динамической компоновки в инженерных расчетах [3]. О.И.Сентябов, Д.Р.Терегулов, Н.И.Морозов расширили прикладные задачи гис "оператора" с использованием динамической библиотеки dll [4]. Е.А.Кудряшов, Д.М.Мельник, А.В.Монаков оптимизировали динамическую загрузку библиотек на архитектуре arm [5].

1.3Цель исследования

Цель исследования – используя язык программирования C создать пользовательскую динамическую библиотеку. Загрузить и вызвать функции компонента в Python проекте.

2 Материалы и методы

Для осуществления поставленных задач используются компиляторы языка C, gcc под linux и MinGW под windows, интерпретатор Python3.8.

3 Результаты и обсуждения

Компонент библиотеки содержит функционал для открытия браузера, который в системе стоит по умолчанию, в параметрах строка должна содержать поисковой запрос. В ниже приведённом примере код будет работать как на платформе windows так и в unix-подобных операционных системах. Пример команды компиляции «gcc -fPIC -shared -o lib1.so lib1.c», где файлы с расширением «.c» - исходный, «.so» - устанавливаемое название библиотеки для linux, в windows используется расширение «.dll» (рис.1).

```c
#include <stdio.h>
#include <stdlib.h>


void open_website(char text[]);

#ifdef _WIN32 //windows
#define FORMAT_STRING "start https://duckduckgo.com/?q=%s"

#else //unix
#define FORMAT_STRING "xdg-open https://duckduckgo.com/?q=%s"
#endif

void open_website(char text[]) {
    size_t len = snprintf(NULL, 0, FORMAT_STRING, text) + 1;
    char *cmd = malloc(len * sizeof(char));
    snprintf(cmd, len, FORMAT_STRING, text);
    system(cmd);
    free(cmd);
}
```

Рис. 1. Исходный код библиотеки

Для взаимодействия с типами данных используемых в языках C/C++ в Python есть специальная библиотека «ctypes». Благодаря данному модулю можно вызывать функции в разделяемых библиотеках, как пользовательских, так и системных.

Для загрузки общих библиотек в windows, в модулье ctypes присутствуют функции windll и oledll, которые автоматически экспортирует имеющиеся библиотеки (рис.2).

```
>>> from ctypes import *
>>> kernel32 = windll.kernel32
>>> print(kernel32)
<WinDLL 'kernel32', handle 75b90000 at 0x28be250>
```

Рис. 3. Подключение основных библиотек windows

В linux, модуль ctypes автоматически экспортирует cdll. Однако экземпляр «ctypes.CDLL» может использоваться и в windows точно также как «ctypes.OleDLL()» и «ctypes.WinDLL()». В linux, для загрузки библиотеки, требуется указать полное имя файла с расширением, в связи с этим доступ к библиотеке как к атрибуту нельзя использовать. Следует загрузить библиотеку, создав экземпляр класса, вызвав конструктор. Так как подключаемая библиотека является пользовательской то в windows, также используется вызов конструктора.

У полученного экземпляра можно вызывать функции из компонента библиотеки как методы объекта класса. В представленной функции в качестве параметра ожидается указатель на изменяемую память. В ctypes есть метод «create_string_buffer», который создает изменяемые блоки памяти различными способами. Префикс «b» определяет строку как однобайтовую последовательность для массива char в языке C (рис.3).

```
import ctypes

lib1 = ctypes.CDLL(r".\lib1.dll")

string = ctypes.create_string_buffer(b'dll')
lib1.open_website(string)
```

Рис. 3. Python скрипт

Приведённый выше скрипт работает как на windows, так и linux. Однако существует возможность, что интерпретатор python под windows при создании экземпляра CDLL может завершить программу с ошибкой «WinError 126», загруженная DLL не найдена. Также необходимо чтобы совпадала разрядность Python и библиотеки (32 или 64 бита).

При выполнении Python программы у пользователя открывается браузер по умолчанию с поисковиком duckduck и поисковым запросом в виде строки переданной в параметре функции (рис. 4).

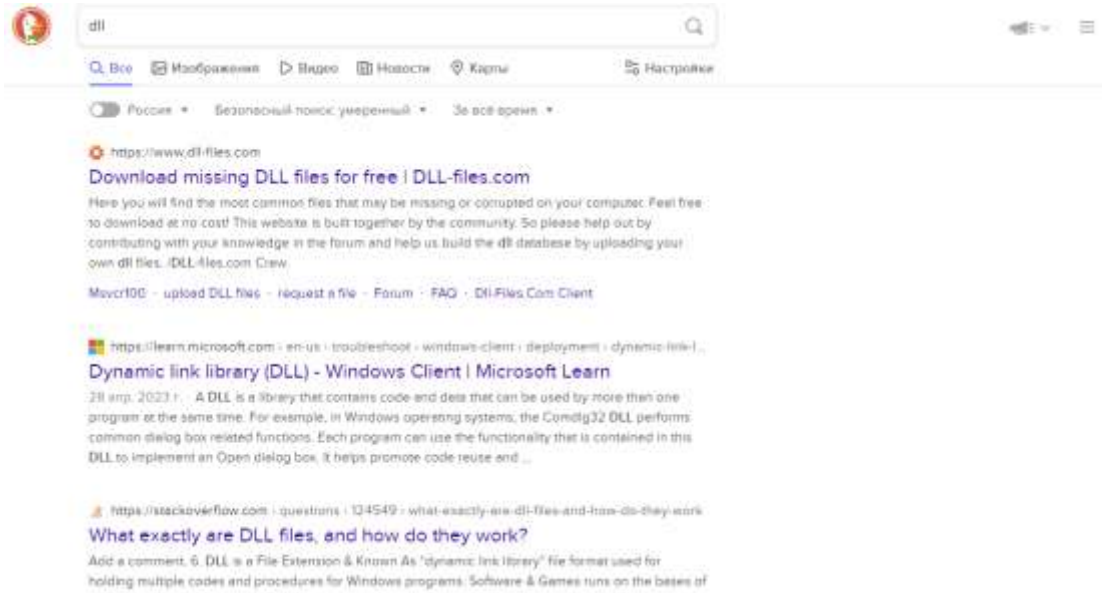

Рис. 4. Результат выполнения функции

Выводы

Таким образом, было описано, как можно создать, подключить и использовать динамические библиотеки, написанные на языке C, в проекте Python на платформах windows и linux.

Библиографический список

1. Попков С.И. Программная реализация межъязыкового взаимодействия на базе динамических библиотек // Нейрокомпьютеры: разработка, применение. 2018. № 3. С. 39-49.
2. Ильичев В.Ю., Жукова Ю.М., Шамов И.В. Вейвлет-анализ непериодических сигналов с использованием специальных библиотек python // Заметки ученого. 2021. № 13. С. 43-47.
3. Хмельнюк М.Г., Важинский Д.И. Создание и использование библиотек динамической компоновки в инженерных расчетах // Холодильная техника и технология. 2016. Т. 52. № 2. С. 93-96.
4. Сентябов О.И., Терегулов Д.Р., Морозов Н.И. Расширение прикладных задач гис "оператора" с использованием динамической библиотеки dll // В сборнике: Информатика: проблемы, методология, технологии. материалы XVI международной научно-методической конференции. 2016. С. 319-323.
5. Кудряшов Е.А., Мельник Д.М., Монаков А.В. Оптимизация динамической загрузки библиотек на архитектуре arm // Труды Института системного программирования РАН. 2016. Т. 28. № 1. С. 63-80.
6. Харченко П.А., Чикалов А.Н. Использование библиотек динамической компоновки для реализации функций api // Труды Северо-Кавказского филиала Московского технического университета связи и информатики. 2012. № 1. С. 155-159.
7. Никишин М.Б., Абросимова С.Н. Основы разработки статистических и динамических библиотек // В сборнике: XLI Огаревские чтения. материалы

научной конференции: в 3-х частях. 2013. С. 235-237.
8. Юрченко А.Н. Сборка статических и динамических библиотек swift с помощью компилятора swift // В сборнике: Электронные системы и технологии. Сборник материалов 57-й научной конференции аспирантов, магистрантов и студентов БГУИР. Редколлегия: Д.В. Лихачевский [и др.]. Минск, 2021. С. 634-636.