# Экстремальный Си

## Глава 3. Объектные файлы

В этой главе подробно обсуждаются различные продукты компиляции проектов на языках C/C++, включая переносимые, исполняемые и разделяемые объектные файлы, а также статические библиотеки.

Эта глава состоит из шести разделов.

1. *Двоичный интерфейс приложений*. Здесь мы поговорим об ABI (application binary interface) и его значимости.
2. *Форматы объектных файлов*. Этот раздел посвящен разным форматам объектных файлов — как актуальным, так и устаревшим. Вы также познакомитесь с самым распространенным форматом объектных файлов в Unix-подобных системах — ELF.
3. *Переносимые объектные файлы*. Здесь будут рассмотрены переносимые объектные файлы и самые первые продукты компиляции проекта на языке C. Мы заглянем внутрь этих файлов (на примере ELF) и посмотрим, что они хранят.
4. *Исполняемые объектные файлы*. В этом разделе мы поговорим об исполняемых объектных файлах и объясним, как они создаются из разных переносимых объектных файлов. Вы увидите различия в их внутренней структуре на примере формата ELF.
5. *Статические библиотеки*. Этот раздел посвящен статическим библиотекам и их созданию. Будет показано, как написать программу, которая использует готовую статическую библиотеку.
6. *Динамические библиотеки*. Здесь речь пойдет о разделяемых объектных файлах. Я продемонстрирую процесс их создания из разных переносимых объектных файлов и покажу, как их использовать в программе. Мы также затронем их внутреннюю структуру на примере формата ELF.

### Двоичный интерфейс приложений

Как вы уже, наверное, знаете, любые библиотеки и фреймворки, независимо от используемых в них технологий и языка написания, предоставляют определенный набор возможностей, известный как *программный интерфейс приложения* (Application Programming Interface, API).

API — своего рода соглашение (или стандарт), принятое двумя программными компонентами, которые обслуживают или используют друг друга. Концепция ABI имеет похожее назначение, только на другом уровне. Если API обеспечивает совместимость двух программных компонентов с точки зрения их функционального взаимодействия, то ABI гарантирует, что две программы и их соответствующие объектные файлы совместимы на уровне машинных инструкций.

### Форматы объектных файлов

### Переносимые объектные файлы

### Исполняемые объектные файлы

### Статические библиотеки

### Динамические библиотеки

### Ручная загрузка разделяемых библиотек

### Резюме

Эта глава в основном посвящена различным типам объектных файлов, которые являются продуктами компиляции проектов на C/C++. Мы рассмотрели следующие темы:

* API и ABI и их отличия;
* разные форматы объектных файлов и краткую историю их появления. Все они произошли от общего предка, но со временем их пути разделились, что в итоге привело к появлению имеющихся на сегодня форматов;
* переносимые объектные файлы и их внутреннюю структуру в контексте формата ELF;
* исполняемые объектные файлы и их отличия от переносимых объектных файлов. Мы также провели краткий обзор исполняемой программы в формате ELF;
* статические и динамические таблицы символов, чтение их содержимого с помощью инструментов командной строки;
* статическую и динамическую компоновку, поиск по разным таблицам символов при создании итогового двоичного файла или запуске программы;
* статические библиотечные файлы. Мы обсудили тот факт, что они фактически являются архивами с рядом переносимых объектных файлов;
* разделяемые объектные файлы (динамические библиотеки). Мы рассмотрели, как их можно собрать из нескольких переносимых объектных файлов;
* позиционно независимый код. Вы узнали, почему переносимые объектные
* файлы, участвующие в создании разделяемой библиотеки, должны быть позиционно независимыми