Как подходить к анализу форматов двоичных файлов

Русский перевод практического руководства Андреаса Пенхака по анализу известных и неизвестных форматов двоичных файлов: строк, чисел, структур, магических чисел, заголовков, выравнивания и эвристического исследования данных.

Больше крутых материалов в моём телеграм-канале @grogrigs

Как подходить к анализу форматов двоичных файлов

Необходимые знания для обратной разработки

Андреас Пехнак

Лето 2015 года

Анализ форматов двоичных файлов

> «Выработайте привычку анализировать: со временем анализ позволит синтезу стать привычкой вашего мышления».
>
> Фрэнк Ллойд Райт

Зачем анализировать форматы двоичных файлов?

Есть множество причин, по которым вас могут интересовать все биты и байты в двоичных файлах. Почти все файлы, которые обычный пользователь компьютера читает и записывает с помощью таких приложений, как текстовые процессоры, программы звукозаписи или видеомонтажа, являются двоичными. Однако специалистам по компьютерам часто требуется копнуть глубже: извлечь, изменить или просто понять содержимое таких файлов.

Экспертам по цифровой криминалистике нередко приходится выявлять в двоичных файлах следы подозрительной деятельности.

Аналитикам вредоносных программ для полного понимания вредоносного ПО необходимо разбирать не только исполняемые файлы.

Повреждённые файлы, например прерванные аудио- или видеозаписи, зачастую можно восстановить лишь при глубоком знании структуры этих файлов.

Программистам, создающим программное обеспечение для записи и чтения двоичных файлов, необходимо хорошо понимать, как их структуры данных представлены в файловом формате. Если что-то работает неправильно, может потребоваться ручной анализ выходных файлов.

Наконец, обратная разработка и анализ двоичных файлов могут просто доставлять удовольствие и, например, позволяют изменять рекорды в сохранённых играх.

Что даст вам эта книга?

Вы узнаете:

• основы форматов двоичных файлов;
• распространённые схемы, встречающиеся во множестве различных форматов;
• как подходить к совершенно неизвестным файловым форматам.

Полезно, если вы уже знаете:

• основы информатики, например что такое бит или байт;
• какой-либо язык программирования, например C или Python.

Если вы уже знакомы с основами форматов двоичных файлов, то можете пропустить первые две главы.

Эта книга посвящена главным образом анализу файлов данных. Сведения о декодировании форматов исполняемых файлов, таких как PE (Portable Executable, Windows), MACH-O (Mac) или ELF (Linux), можно найти в различных источниках в Интернете.

Основы

> «Простые решения редко бывают простыми. Чтобы взяться за анализ очевидного, нужен весьма необычный ум».
>
> Альфред Норт Уайтхед

Введение

Любая информация, которую прикладная программа хочет постоянно хранить в файле, должна иметь некоторое чётко определённое представление. И вся эта информация должна записываться по определённым правилам, чтобы впоследствии данные можно было прочитать обратно в память.

Для вас это означает, что анализ двоичного файла в основном сводится к пониманию структуры файла и смысла определённой информации.

Хотя двоичные файлы прежде всего состоят из байтов, на уровне файлового формата вы главным образом встретите текст и числа, часто объединённые в структуры. Здесь мы изучим атомы, а позднее соединим их в молекулы.

Текстовые строки

Кодировка символов

Поскольку компьютеры умеют обрабатывать только числа, каждому символу необходимо поставить в соответствие некоторое числовое представление. Такое сопоставление также называется кодировкой текста. Большинство кодировок основано на US-ASCII с её 128 определениями (7 бит). Для хранения текстов на таких языках, как французский или русский, были изобретены так называемые кодовые страницы, использующие вторые 128 значений, которые можно представить одним байтом.

Позднее появился Unicode, определяющий более 120 000 символов. Кроме того, он позволяет менять направление чтения для таких языков, как иврит или арабский. Помните, что текст Unicode может быть представлен в файле разными способами. Очень распространена кодировка UTF-8, использующая до 6 байт на символ, однако в различных файловых форматах также встречается UTF-16.

В файловых форматах, возникших в мире IBM, часто используется текст в кодировке EBCDIC. В отличие от ASCII, EBCDIC не определяет символы в последовательном порядке, из-за чего программное обеспечение, скомпилированное на машинах на основе EBCDIC, нередко работает неправильно.

Хранение

Помимо различных кодировок символов существует несколько распространённых способов задавать длину текста в файле. Во многих случаях представление в файле совпадает с

представлением в оперативной памяти.

Фиксированная длина

Во многих файловых форматах максимальная длина строк фиксирована. Обычно оставшееся после сохраняемого текста место заполняется нулевыми байтами.

Строка длиной 22 байта, содержащая 16 символов

S	**o**	**m**	**e**	**пробел**	**s**	**a**	**m**	**p**	**l**	**e**	**пробел**	**t**	**e**	**x**	**t**	`\0`	`\0`	`\0`	`\0`	`\0`	`\0`

С завершающим нулём

Поскольку C уже более 40 лет является очень популярным языком программирования, многие файловые форматы содержат строки в том виде, в котором они хранятся в оперативной памяти программами, разработанными на C. Конец таких строк обозначается одним нулевым байтом или двумя нулевыми байтами для двухбайтовых кодировок символов.

Строка длиной 17 байт, содержащая 16 символов

S	**o**	**m**	**e**	**пробел**	**s**	**a**	**m**	**p**	**l**	**e**	**пробел**	**t**	**e**	**x**	**t**	`\0`

Pascal

Такие языки программирования, как Pascal, Modula или Delphi, хранят длину строки в её первом байте. Поэтому строки в файловых форматах, записанных таким программным обеспечением, часто следуют тому же соглашению. Первый байт строки содержит число следующих за ним символов.

Строка Pascal, у которой первый байт содержит количество следующих за ним символов

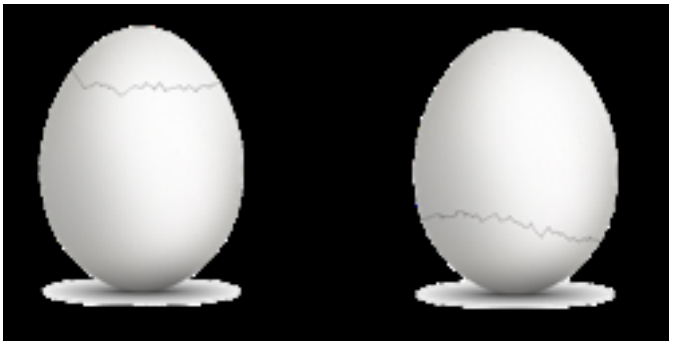
16	**S**	**o**	**m**	**e**	**пробел**	**s**	**a**	**m**	**p**	**l**	**e**	**пробел**	**t**	**e**	**x**	**t**

Числа

Хотя текстовые строки играют значительную роль в двоичных файлах, числа ещё важнее. Они используются не только для представления непосредственно исчисляемых сущностей, но также могут обозначать перечисления или маски.

Порядок байтов (эндианность)

При работе с форматами двоичных файлов вам совершенно необходимо знать понятие порядка байтов. Этот термин связан с сюжетом романа Джонатана Свифта «Путешествия Гулливера».



Для наших целей достаточно помнить, что в некоторых файловых форматах порядок байтов чисел меняется на обратный (Little Endian). В основном это относится к форматам, созданным на компьютерах с процессорами Intel.

Учтите, что при программировании на языке высокого уровня это обычно не играет роли. Однако если посмотреть дамп памяти числовых переменных, можно заметить обратный порядок байтов.

Некоторые файловые форматы, например TIFF, могут содержать числа как с малым, так и с большим порядком байтов в зависимости от определённого индикатора в файле.

Пример 32-битного, или 4-байтового, числа:

Порядок байтов	`0x12345678` **(десятичное 305419896)**	`0x87654321` **(десятичное 2271560481)**
Big Endian (большой порядок байтов)	`12 34 56 78`	`87 65 43 21`
Little Endian (малый порядок байтов)	`78 56 34 12`	`21 43 65 87`

> **Важно.** Обращается порядок байтов, а не порядок битов!

Целые числа

При работе с целым числом необходимо учитывать главным образом два различия: его размер и то, является ли оно знаковым или беззнаковым.

Типичные размеры целых чисел составляют 1, 2, 4 и 8 байтов. Форматы, рассчитанные на уменьшение размера файлов, часто содержат числа произвольной битовой длины.

Смещения

В форматах двоичных файлов часто встречаются числа, интерпретируемые как смещения в файле. Такое число можно считать указателем на определённую позицию в файле, с которой следует продолжить разбор.

Существуют абсолютные и относительные смещения. Абсолютное смещение отсчитывается от начала файла. Относительные смещения прибавляются к некоторой другой позиции в файле.

Числа с плавающей точкой

Чтобы стало возможно представлять очень малые и очень большие величины, были изобретены числа с плавающей точкой. Почти всегда они соответствуют стандарту IEEE 754 и чаще всего имеют размер 4 или 8 байтов. Числа половинной точности (2 байта) или четверной точности (16 байтов) редко встречаются в двоичных файлах.

Флаги

Флаги обычно используются, чтобы показать, включена или выключена некоторая «возможность». На уровне файла это означает, что один бит представляет значение флага: ноль означает «выключено» или «ложь», а единица - «включено» или «истина». Часто несколько связанных флагов объединяют в одном числовом значении.

Пример флагов стиля окна в Windows:

Имя	Маска (шестнадцатеричная)	Описание
`WS_EX_ACCEPTFILES`	`0x00000010`	Окно принимает файлы, перетаскиваемые в него.
`WS_EX_CLIENTEDGE`	`0x00000200`	Окно имеет утопленную границу.
`WS_EX_CONTEXTHELP`	`0x00000400`	Строка заголовка окна содержит знак вопроса.
`WS_EX_APPWINDOW`	`0x00040000`	Заставляет отображать видимое окно верхнего уровня на панели задач.

Чтобы проверить, равен ли бит флага в числе нулю или единице, применяют маски. Маска - это число, в котором установлен проверяемый бит или биты. Если результат выполнения двоичной операции AND над маской и значением больше нуля, флаг установлен.

Значения масок также можно использовать для установки отдельных битов в числе. Для этого достаточно выполнить двоичную операцию OR над маской и значением.

Флаг	Маска (двоичная)	Маска (шестнадцатеричная)	Маска (десятичная)
1	`00000001`	`0x01`	$1 = 2^0$
2	`00000010`	`0x02`	$2 = 2^1$
3	`00000100`	`0x04`	$4 = 2^2$
4	`00001000`	`0x08`	$8 = 2^3$
5	`00010000`	`0x10`	$16 = 2^4$
6	`00100000`	`0x20`	$32 = 2^5$
7	`01000000`	`0x40`	$64 = 2^6$

Флаг	Маска (двоичная)	Маска (шестнадцатеричная)	Маска (десятичная)
8	`10000000`	`0x80`	$128 = 2^7$

Значение (двоичное)	Маска (двоичная)	Результат AND	Результат OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Структуры

В языках программирования структурами называются именованные совокупности таких элементов, как числа, строки или другие объекты. На уровне файла всё выглядит несколько иначе. Поскольку независимой метки с именем нет, структуры приходится распознавать по их позиции или по некоторому элементу внутри структуры.

Выравнивание и заполнение

В таких языках программирования, как C, элементы структур по умолчанию выравниваются по позициям байтов, кратным 2, 4, 8 или 16 в зависимости от архитектуры процессора. Подобным образом в файловых форматах структуры или их элементы иногда выравниваются по границе, кратной 2 или 4 байтам.

Выравнивание структур встречается, например, в файловых форматах IFF85 и TrueType. Обычно область заполнения перед выровненной структурой заполняется байтами `0x00`.

Форматы двоичных файлов

> «Величайшие мгновения наступают, когда результат появляется на графике или в статистическом анализе: в тот миг вы осознаёте, что знаете нечто, чего не знает никто другой, и получаете удовольствие, размышляя, как об этом рассказать».
>
> Эмили Остер

Типичные схемы

Если рассмотреть множество форматов двоичных файлов, окажется, что многие из них основаны на некоторых общих принципах.

Всегда помните:

• содержимое файла читается в определённом порядке;
• чем гибче файловый формат, тем больше в нём подсказок о том, как его читать.

Простейший возможный файловый формат содержит только элементы (числа, строки и так далее), расположенные в фиксированных позициях файла. Если, например, некоторый элемент необязателен, в файле должна присутствовать дополнительная информация, указывающая, следует ли его читать.

То же относится к альтернативным структурам: внутри структур или вне их должна находиться какая-либо подсказка о том, какую структуру следует прочитать.

Опознавательные признаки и магические числа

Чтобы тип файла можно было определить не только по расширению его имени, многие файловые форматы начинаются с уникальной последовательности байтов. Команда UNIX `file` способна определять формат файлов главным образом по магическим числам.

Формат	**Текст (ASCII)**	**Байты**
GIF	`GIF89a`	`47 49 46 38 39 61`
PDF	`%PDF`	`25 50 44 46`
ZIP	`PK`	`50 4B`

Заголовки файлов

Большинство форматов двоичных файлов начинается с некоторой структуры, называемой заголовком файла. В зависимости от спецификации магическое число может входить в заголовок либо располагаться перед ним.

Обычно заголовки файлов содержат номера версий, файловые смещения других структур в файле, ширину и высоту изображения или, в общем случае, сведения, необходимые для чтения остальной части файла.

В некоторых файловых форматах программа записи не всегда заранее знает, что именно будет записано в файл. В таких форматах в конце файла может находиться структура, обеспечивающая доступ к его содержимому. Примерами служат ZIP и PDF.

Длины структур

В зависимости от содержащихся в структуре данных она может иметь фиксированную или переменную длину в байтах.

Типы длины структуры:

Длина структуры	**Описание**
Фиксированная	Длина не указана в файле, но известна приложению, которое его читает.
Элемент длины	Внутри структуры или вне её, например в заголовке файла, длина структуры хранится в числовом элементе.
Разделитель	Сама структура или её элементы продолжаются до появления определённой последовательности байтов. Типичный пример такого случая - строки с завершающим нулём.

Альтернативные структуры

Предположим, гипотетический файловый формат может хранить структуры точек, линий и треугольников в любом порядке. Помимо самих координат программа чтения должна получить некоторую подсказку о том, какой объект следует читать следующим.

Порядок	**Объект**	**Содержимое структуры**
1	Треугольник	`x1, y1, x2, y2, x3, y3`
2	Линия	`x1, y1, x2, y2`
3	Треугольник	`x1, y1, x2, y2, x3, y3`
4	Точка	`x1, y1`
5	Треугольник	`x1, y1, x2, y2, x3, y3`
6	Линия	`x1, y1, x2, y2`

Порядок	Объект	Содержимое структуры
7	Точка	`x1, y1`

Двоичное представление этих данных могло бы выглядеть следующим образом, если предположить, что каждое число помещается в один байт:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	03	x1	y1	x2	y2	x3	y3	02	x1	y1	x2	y2	03	x1	y1	x2
1	y2	x3	y3	01	x1	y1	03	x1	y1	x2	y2	x3	y3	02	02	x1
2	y1	x2	y2	01	x1	y1										

В этом случае дополнительно указывать длины структур не требуется: треугольник всегда состоит из трёх точек, а линия - из двух.

Неизвестные форматы

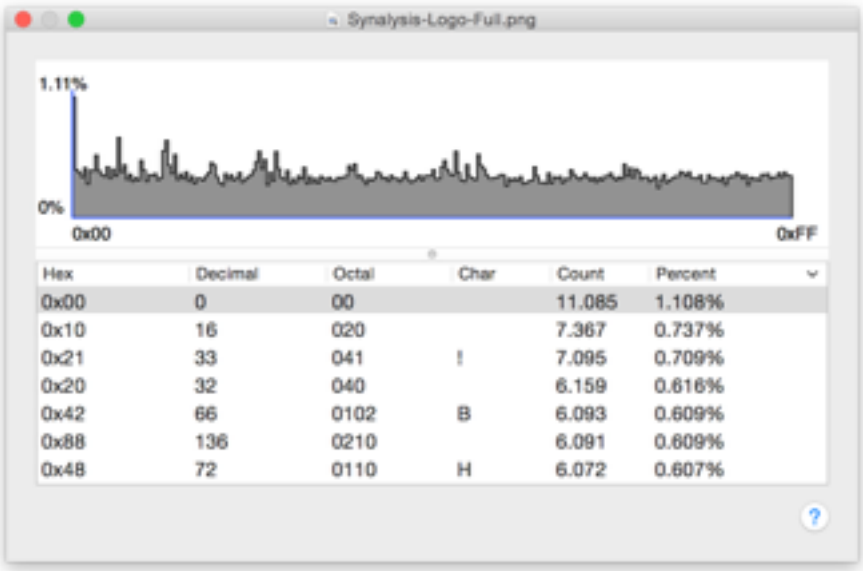
> «Мой ум восстаёт против бездействия. Дайте мне проблемы, дайте работу, предложите самую непостижимую криптограмму или самый замысловатый анализ, и я окажусь в своей родной стихии. Но я ненавижу унылую рутину существования. Я жажду умственного подъёма».
>
> Артур Конан Дойл

С чего начать

Предположим, вы хотите декодировать двоичный файл, но у вас нет спецификации. Здесь я предлагаю несколько простых шагов, которые помогут больше узнать о таком файле. Пока что человеческий мозг по-прежнему остаётся одним из лучших инструментов для выявления закономерностей и осмысления их значения.

Гистограмма

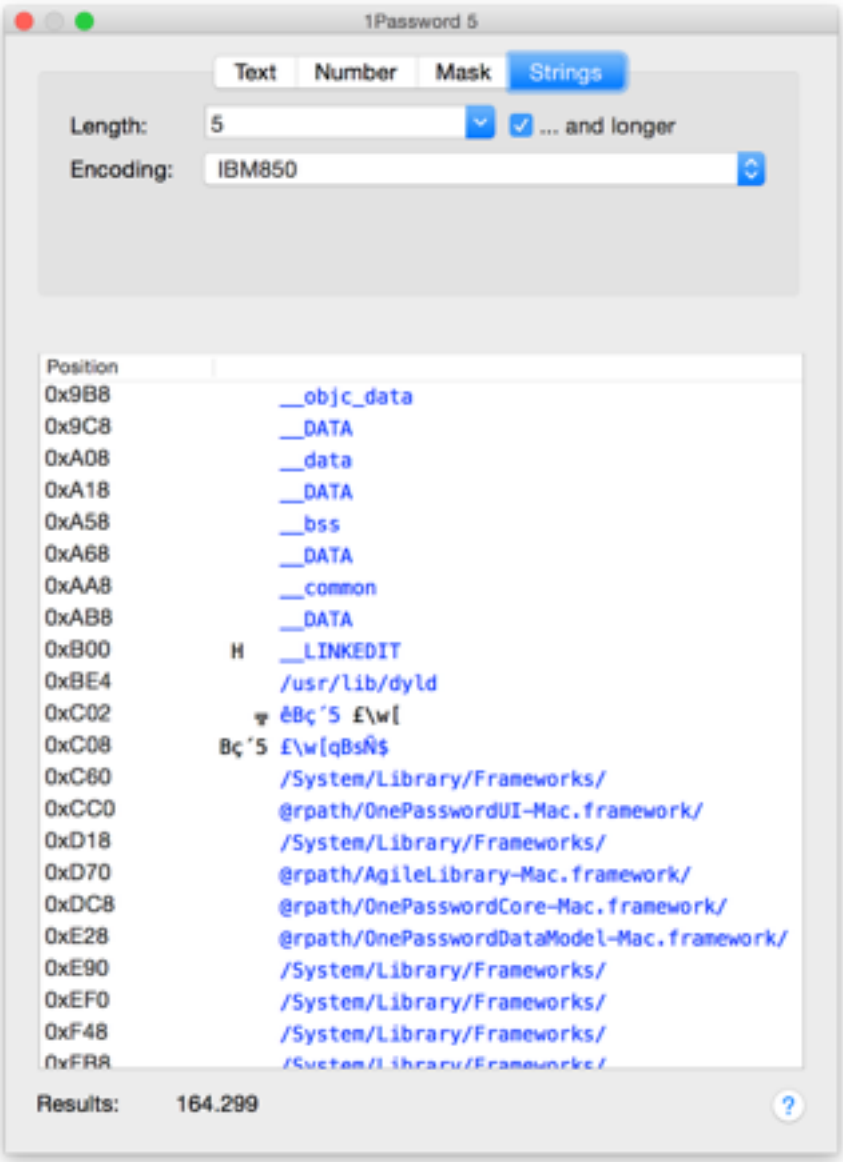
Удобным средством для изучения характеристик файла служит гистограмма. По существу, она показывает, сколько раз каждое значение байта от 0 до 255 встречается в файле.



Что гистограмма может рассказать о файле? Во-первых, очень равномерное распределение количества байтов указывает на то, что файл сжат или зашифрован. Во-вторых, отдельные пики подсказывают, какие байты играют важную роль в файловом формате. Пик байтов `0x00` часто можно не учитывать, поскольку ими дополняется большинство строк и других данных.

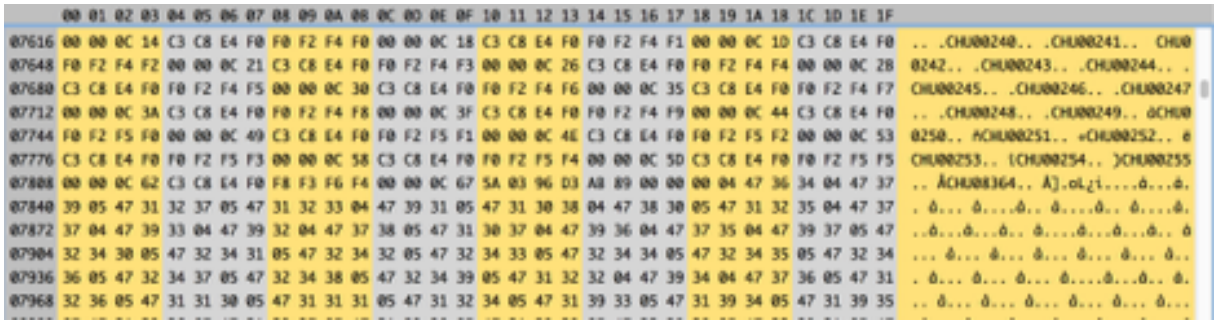
Строки

Ещё один полезный инструмент, позволяющий составить представление о структуре файла, - `strings`. Можно воспользоваться версией для командной строки в Unix или программой с графическим интерфейсом, например Hexinator либо Synalyze It!



Разделы

Многие файловые форматы состоят из различных разделов. Чаще всего их можно распознать, быстро прокручивая файл в шестнадцатеричном редакторе и просматривая текстовый столбец.



Приложение, которому «известен» файловый формат, может читать эти разделы тремя способами:

• разделы начинаются в фиксированных позициях, которые всегда одинаковы во всех файлах данного формата;
• на начало разделов указывают файловые смещения, часто хранящиеся в заголовке файла;
• известна длина всех предшествующих разделов.

Если записать начальные позиции и длины всех разделов, которые удалось распознать визуально, позднее можно попытаться найти эти сведения в других местах файла.

Заголовок файла

Поскольку большинство файловых форматов содержит некоторый заголовок, отмечающий начальную точку для декодирования остальной части файла, именно здесь удобно искать отмеченные ранее файловые смещения. В большинстве случаев файловые смещения хранятся в виде 4-байтовых чисел, а в более новых форматах числа могут занимать и 8 байтов.

Попытайтесь также определить другие сведения, о которых вам известно, что они хранятся в файле. Например, если вы знаете, что файл содержит изображение определённой ширины и высоты, попробуйте найти эти значения. Помните, что числа могут храниться в обратном порядке байтов (Little Endian).

Структуры

Внимательнее рассмотрев раздел файла, можно заметить, что он состоит из последовательности похожих структур. Часто эти структуры следуют некоторой простой схеме и начинаются с идентификатора и поля длины, содержащего длину структуры в байтах.

В некоторых случаях присутствует лишь идентификатор типа структуры, а длина явно не указывается, поскольку структура этого типа всегда имеет одну и ту же длину.

Тип структуры	Длина структуры	Данные

Иногда поле длины содержит длину всей структуры, а иногда - только следующих за ним данных, поэтому не ищите точных чисел.

Вот как можно легко определить размер структур фиксированной длины. В шестнадцатеричном редакторе, позволяющем задавать произвольное число столбцов, просто изменяйте это число, пока одинаковые схемы в шестнадцатеричном или текстовом представлении не выровняются по вертикали.

Теперь длину структуры можно легко прочитать в строке заголовка шестнадцатеричного представления.