

 **PE Linker**

* COFF - формат

- * Common Object File Format - стандартный формат объектного файла
- * Некоторые поля файла имеют восьмеричный формат
- * COFF-формат был сам по себе неплохой отправной точкой, но нуждался в расширении, чтобы удовлетворить потребностям новых операционных систем, таких как Windows NT или Windows 98. Результатом такого усовершенствования явился PE-формат

* **Portable Executable - переносимый исполняемый**

- * Это формат исполняемых файлов, объектного кода и динамических библиотек, используемый в 32- и 64-битных версиях операционной системы Microsoft Windows.
- * Формат PE представляет собой структуру данных, содержащую всю информацию, необходимую PE загрузчику для проецирования файла в память.
- * PE-файл состоит из заголовка и некоторого набора секций, количество и размер которых зависит от информации, содержащейся в заголовке.

* COFF и PE. В чем различие?

- * Компоновщик не превращает объектный файл в исполняемый, а создаёт загрузочный модуль на основе информации, содержащейся в одном или нескольких объектных модулях.
- * Другими словами, объектный и исполняемый файлы - это два совершенно разных файла, хотя и содержащие значительный объем одинаковой информации.

Форматы исполняемых файлов							
Windows, DOS, OS/2	.COM	.EXE(MZ/NE/LE/LX/PE)					
Unix	a.out	COFF	ECOFF	ELF	Mach-O	SOM	XCOFF
Прочие	Intel HEX	PEF	SREC				

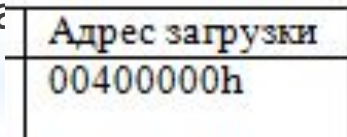
* Шаг 1 - Написание программы

```
* .386
* .model flat
*     extrn MessageBoxA: dword
*     extrn ExitProcess: dword
*     extrn GetComputerNameA: dword
* .code
* _start:
*     push offset nSize
*     push offset lpBuffer
*     call GetComputerNameA
*     push 40h
*     push offset msg
*     push offset lpBuffer
*     push 00h
*     call MessageBoxA
*     push 00h
*     call ExitProcess
* .data
*     lpBuffer db 20 dup(0),0
*     nSize db 3 dup(0),14h
*     msg db 'PE Linker',0
* end start
```

* обязательно должна использоваться модель памяти FLAT (плоская бесsegmentная модель).

* все внешние функции (в данном случае - функции API) необходимо объявлять с помощью директивы:
extrn <имя функции>: dword

* имена функций чувствительны к регистру символов!!!

*  дать из задания!

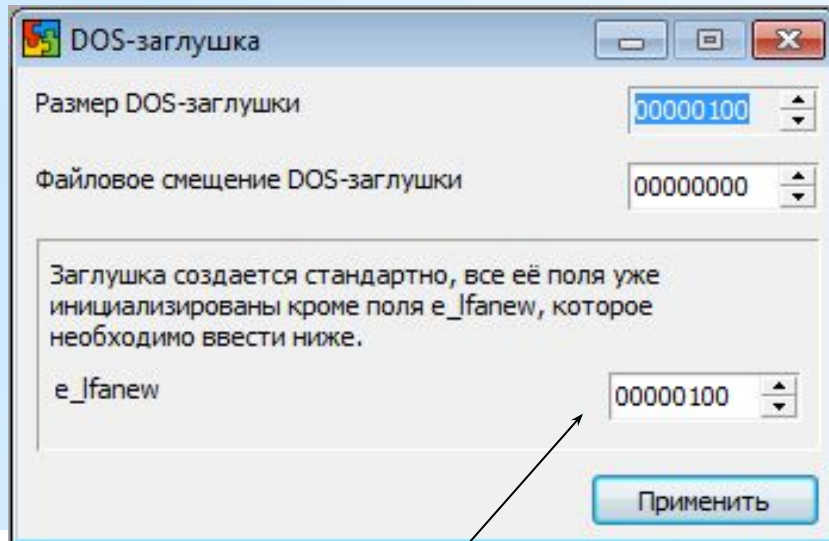
* После написания программы её необходимо откомпилировать с помощью команды:

ml/coff /c <имя файла>

* Шаг 2 - Создание заголовка PE-файла

- * Как и в других исполняемых форматах от Microsoft, заголовки не находятся в самом начале файла. Вместо этого несколько сотен первых байтов типичного PE-файла заняты под *заглушку DOS*.
- * Эта заглушка представляет собой минимальную DOS-программу, которая выводит что-либо вроде: "Эта программа не может быть запущена под DOS".
- * Все это предусматривает случай, когда пользователь запускает программу Win32 в среде, которая не поддерживает Win32, получая при этом приведенное выше сообщение об ошибке.

* Шаг 2 - Создание заголовка PE-файла

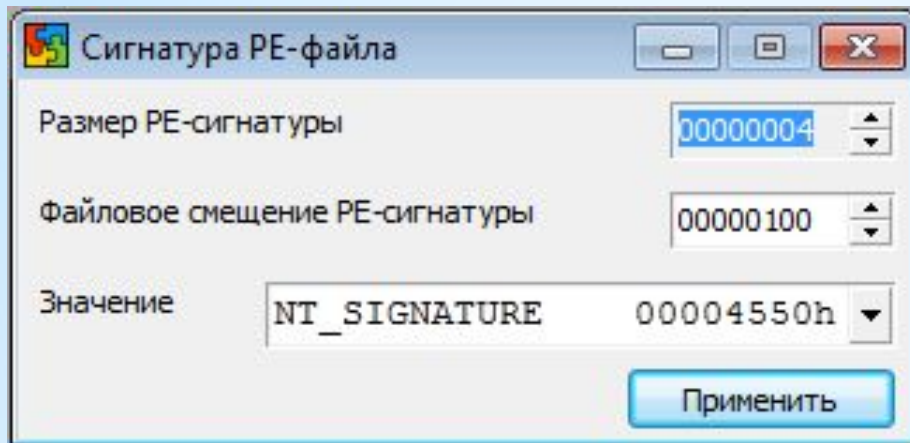


* первый байт отображения файла соответствует первому байту заглушки DOS.

* настоящий заголовок можно обнаружить, найдя его стартовое смещение, которое хранится в заголовке DOS.

* Поле `e_lfanew` собственно и содержит относительное смещение (RVA) настоящего заголовка PE-файла.

* Шаг 2 - Создание заголовка PE-файла



+ = Файловое смещение заголовка

* Поле Signature (сигнатура - подпись), представленное как ASCII код, - это PE00 (два нулевых байта после PE).

* Шаг 2 - Создание заголовка PE-файла

Файловый заголовок

Размер файлового заголовка: 00000014

Файловое смещение заголовка: 00000104

Machine: IA-32

NumberOfSections: 0003

TimeDateStamp: 00000000

PointerToSymbolTable

NumberOfSymbols

SizeOfOptionalHeader

Characteristics



Файловое смещение
дополнительного
заголовка

* NumberOfSections - кол-во
секций = 3 (кода, данных,

Characteristics

- ☒ Файл не содержит перемещений
- ☒ Файл представляет исполнимое изображение EXE
- ☒ Файл не содержит номеров строк
- ☒ Файл не содержит локальных символов
- ☐ Установлен агрессивный режим работы
- ☐ Приложение может занимать более 2-х Гб
- ☐ Байты в машинном слове переставлены (LO)
- ☒ В машинном слове 32 бита
- ☐ Отладочная информация в DBG файле
- ☐ Если изображение на съемном носителе, то скопировать и запустить из SWAP файла
- ☐ Если изображение в сети, то скопировать и запустить из SWAP файла
- ☐ Системный файл
- ☐ Файл является библиотекой динамической компоновки
- ☐ Файл должен быть запущен только на "мощной" машине
- ☐ Байты в машинном слове переставлены (HI)

* Шаг 2 - Создание заголовка PE-файла

Дополнительный заголовок

Размер дополнительного заголовка: 00E0

Файловое смещение заголовка: 00000118

Magic	010B	SubsystemVer	0004.0000
LinkerVer	02.23	Reserved	00000000
SizeOfCode	00000000	SizeOfImage	00004000
SizeOfInitializedData	00000000	SizeOfHeaders	00000400
SizeOfUninitializedData	00000000	CheckSum	00000000
AddressOfEntryPoint	00001000	Subsystem	windows_gui
BaseOfCode	00000000	DLLCharacteristics	00000000
BaseOfData	00000000	SizeOfStackReserve	00100000
ImageBase	00400000	SizeOfStackCommit	00001000
SectionAlignment	00001000	SizeOfHeapReserve	00100000
FileAlignment	00000200	SizeOfHeapCommit	00001000
OSVer	0004.0000	LoaderFlags	00000000
ImageVer	0000.0000	NumberOfRVAandSizes	00000010
ImageDataDirectory		VirtualAddress	00000000
0 Export Directory		Size	00000000
1 Import Directory			
2 Resource Directory			

Применить

Файловое смещение таблицы секций

+ =

* ImageBase - адрес загрузки (см. шаг 1)

* Magic - слово-сигнатура, определяющее состояние отображенного файла (010b - исполняемое отображение). Для 64 разрядной системы равно 020b.

* Шаг 2 - Создание заголовка PE-файла

- * $\text{AddressOfEntryPoint} = 1000$ (входная точка главного потока = RVA данных секции кода(.text))
- * $\text{SectionAlignment} \geq 1000$ (const Кратность выравнивания секций в памяти = размер страницы)
- * $\text{FileAlignment} \geq 200$ (const Кратность выравнивания секций на диске = размер сектора винчестера)
- * $\text{SizeOfImage} = \text{VirtualAddress}(\text{последней секции}) + \text{VirtualSize}(\text{последней секции}) = 3000 + 1000 = 4000$
- * $\text{SizeOfHeaders} = 400$ (const = размер всех заголовков и таблицы секций)

* Шаг 2 - Создание заголовка PE-файла

- * `SizeOfStackReserve = 100000` (const = зарезервированный в вирт. пространстве объём для стека главного потока)
- * `SizeOfStackCommit = 1000` (const = зарезервированный в пространстве физ. памяти объём для стека главного потока)
- * `SizeOfHeapReserve = 100000` (const = зарезервированный объём для главного хипа)
- * `SizeOfHeapCommit = 1000` (const = зарезервированный в пространстве физ. памяти объём для главного хипа)

* Шаг 3 - Создание секций PE-файла

Заголовок 1-й секции «.text»

Размер таблицы секций: 00000078

Файловое смещение таблицы секций: 000001F8

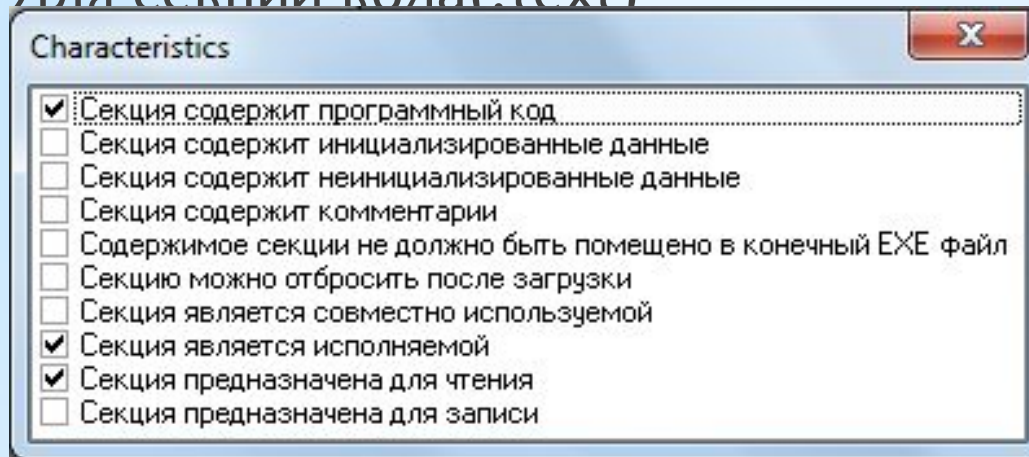
Name	.text
VirtualSize	00001000
VirtualAddress	00001000
SizeOfRawData	00000200
PointerToRawData	00000400
PointerToRelocations	00000000
PointerToLinenumbers	00000000
NumberOfRelocations	0000
NumberOfLinenumbers	0000
Characteristics	60000020

Применить

- * Name - название секции
- * VirtualSize = 1000 (вирт. размер секции)
- * VirtualAddress = 1000 + VirtualSize * (номер секции - 1) (адрес начала секции в памяти)
- * SizeOfRawData = 200 (физ. размер секции)
- * PointerToRawData = 400 + SizeOfRawData * номер секции (смещение относительно начала файла)

* Шаг 3 - Создание секций PE-файла

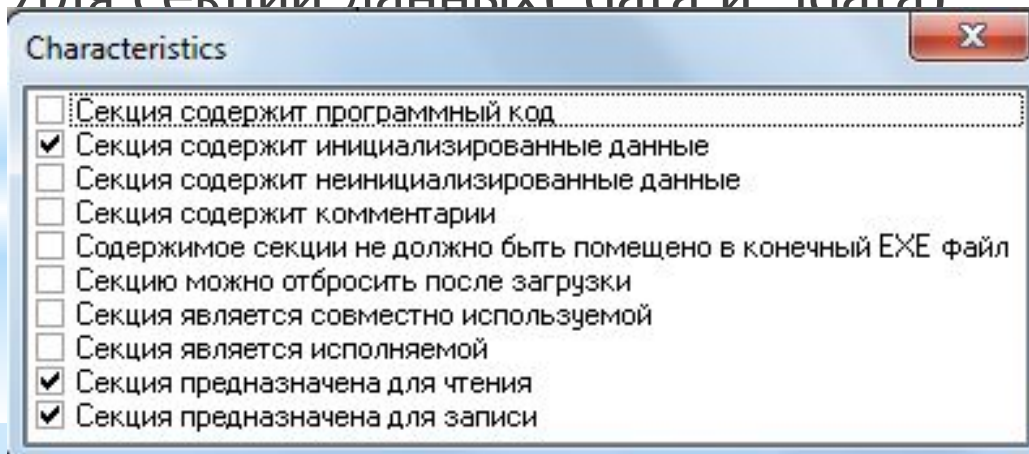
* Для секции кода(.text)



The screenshot shows a 'Characteristics' dialog box with a list of checkboxes. The first checkbox, 'Секция содержит программный код', is checked. The other checkboxes are: 'Секция содержит инициализированные данные', 'Секция содержит неинициализированные данные', 'Секция содержит комментарии', 'Содержимое секции не должно быть помещено в конечный EXE файл', 'Секцию можно отбросить после загрузки', 'Секция является совместно используемой', 'Секция является исполняемой', 'Секция предназначена для чтения', and 'Секция предназначена для записи'.

Свойство	Выбрано
Секция содержит программный код	Да
Секция содержит инициализированные данные	Нет
Секция содержит неинициализированные данные	Нет
Секция содержит комментарии	Нет
Содержимое секции не должно быть помещено в конечный EXE файл	Нет
Секцию можно отбросить после загрузки	Нет
Секция является совместно используемой	Нет
Секция является исполняемой	Да
Секция предназначена для чтения	Да
Секция предназначена для записи	Нет

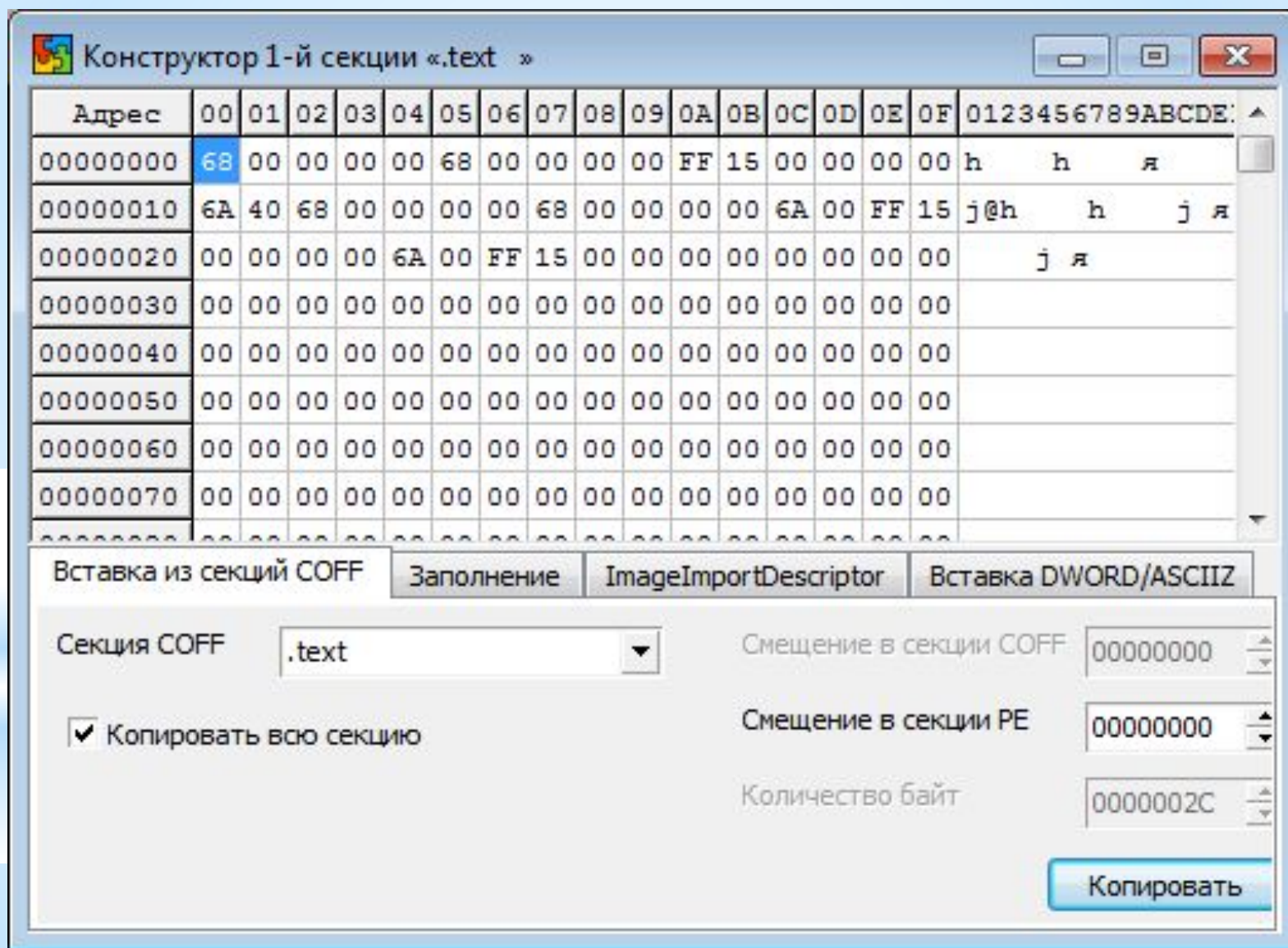
* Для секции данных(data и idata)



The screenshot shows a 'Characteristics' dialog box with a list of checkboxes. The second checkbox, 'Секция содержит инициализированные данные', is checked. The other checkboxes are: 'Секция содержит программный код', 'Секция содержит неинициализированные данные', 'Секция содержит комментарии', 'Содержимое секции не должно быть помещено в конечный EXE файл', 'Секцию можно отбросить после загрузки', 'Секция является совместно используемой', 'Секция является исполняемой', 'Секция предназначена для чтения', and 'Секция предназначена для записи'.

Свойство	Выбрано
Секция содержит программный код	Нет
Секция содержит инициализированные данные	Да
Секция содержит неинициализированные данные	Нет
Секция содержит комментарии	Нет
Содержимое секции не должно быть помещено в конечный EXE файл	Нет
Секцию можно отбросить после загрузки	Нет
Секция является совместно используемой	Нет
Секция является исполняемой	Нет
Секция предназначена для чтения	Да
Секция предназначена для записи	Да

* Шаг 3 - Создание секций PE-файла



* Шаг 3 - Создание секций PE-файла

*Если это не секция “.idata” то

- 1) Клик мышкой на ячейку (0;0)
- 2) В нижней части всплывшего окна выбираем вкладку «Вставка из секции COFF»
- 3) устанавливаем в поле «Секция COFF» открывшейся панели имя совпадающее с именем этой секции.
- 4) устанавливаем в поле «Копировать всю секцию» открывшейся панели галочку.
- 5) Нажимаем кнопку «Копировать».

* Шаг 3 - Создание секций PE-файла

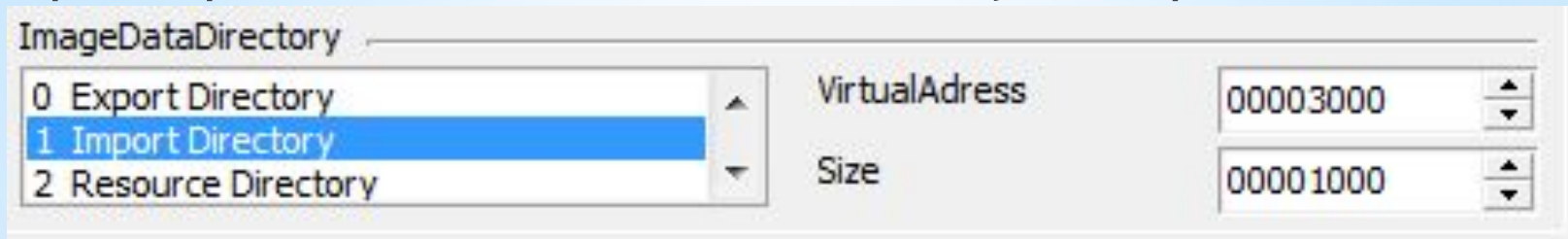
* Что храниться в секции “.idata” ?

- Перед загрузкой в память информация, хранящаяся в секции .idata PE-файла, содержит информацию, необходимую для того, чтобы загрузчик мог определить адреса целевых функций и пристыковать их к отображению исполняемого файла.
- После загрузки секция .idata содержит указатели функций, импортируемых EXE-файлом или DLL.

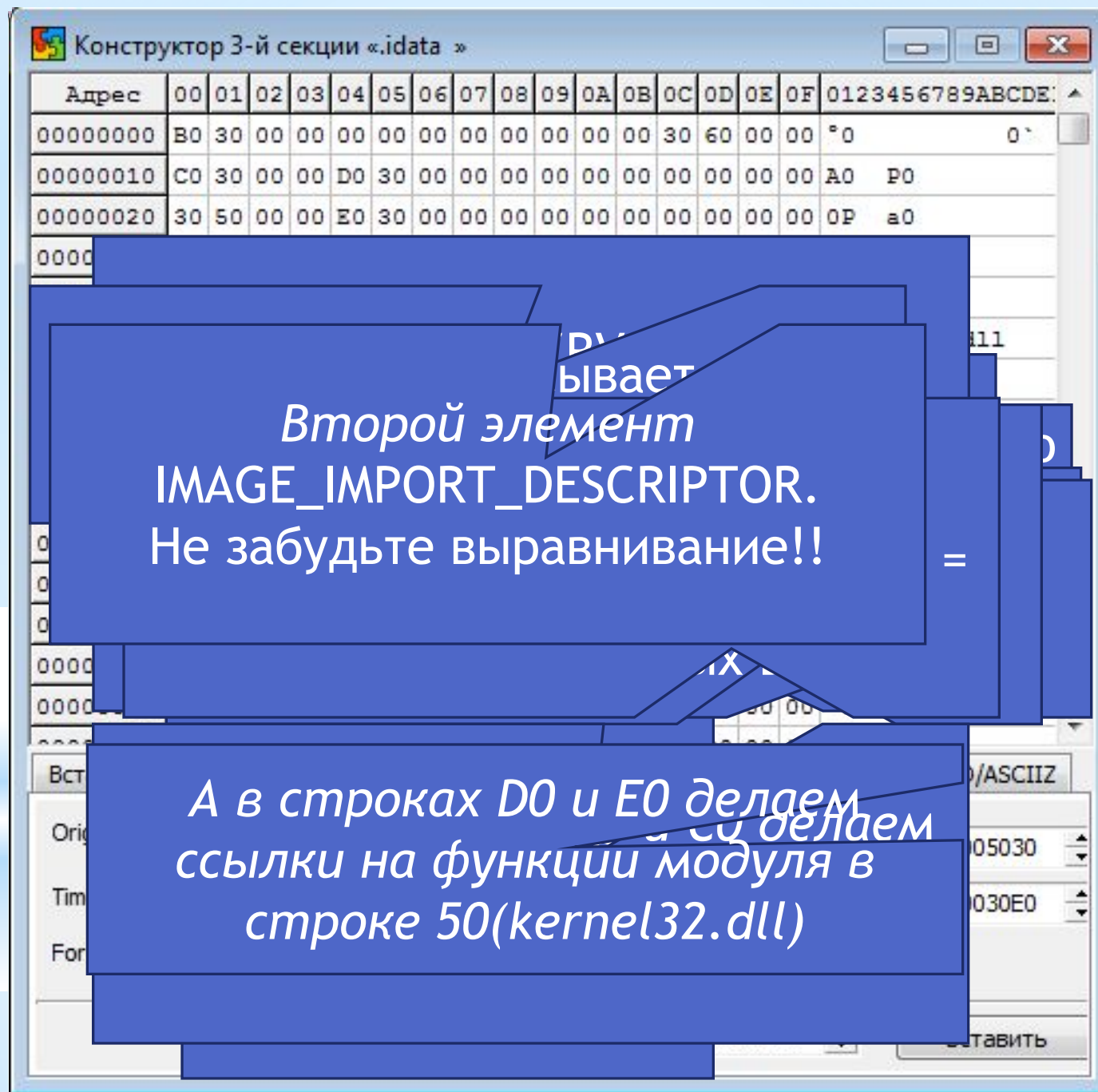
* Шаг 3 - Создание секций PE-файла

* Если это секция “.idata” то

* Предварительно подключаем таблицу импорта в заголовке.



ImageDataDirectory	
0 Export Directory	VirtualAddress: 00003000
1 Import Directory	Size: 00001000
2 Resource Directory	



* Шаг 4 - Разрешение статических и внешних ссылок

- * Очень важный этап компоновки - разрешение статических и внешних ссылок.
- * На этапе компиляции неизвестны реальные адреса переменных и функций API, поэтому компилятор превращает адреса переменных в статические, а адреса функций API - во внешние ссылки.
- * Информация о неразрешенных ссылках хранится в двух местах в объектном модуле: в COFF-таблице символов и в списках привязок для каждой секции.

* Шаг 4 - Разрешение статических и внешних ссылок

Для разрешения ссылок для каждой секции COFF-файла используется следующий алгоритм:

- 1) найти первую, еще не разрешенную ссылку в списке привязок данной секции. Если таких нет, то алгоритм завершен;
- 2) найти символ в COFF-таблице, на который ссылается данная привязка;
- 3) если символ является внешним (тип EXTERNAL), то перейти к пункту 9;
- 4) если данный символ имеет тип STATIC, то данная ссылка является разрешимой;
- 5) найти секцию PE, соответствующую секции с номером SectionNumber COFF-Файла;
- 6) сосчитать неизвестный адрес по следующей формуле:

$$\begin{aligned} \text{Искомый_адрес} = & \text{Адрес_загрузки(см. шаг 1)} + \\ & \text{RVA_секции_из_пункта_5_алгоритма} + \\ & \text{Поле_Value_из_COFF-символа} \end{aligned}$$

* Шаг 4 - Разрешение статических и внешних ссылок

7) в секции PE-файла, соответствующей данной секции COFF-файла, по смещению Address из привязки вставить значение, полученное в пункте 6 алгоритма;

8) перейти к пункту 1.

9) сосчитать неизвестный адрес по следующей формуле:

Искомый_адрес = Адрес_загрузки + RVA элемента массива FirstThunk описывающего данную функцию

10) в секции PE-файла, соответствующей данной секции COFF-файла, по смещению Address из привязки вставить значение, полученное в пункте 5 алгоритма;

11) перейти к пункту 1.

* Шаг 4 - Разрешение статических и внешних ссылок

Конструктор 1-й секции «.text»

Адрес	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDE
00000000	68	15	20	40	00	68	00	20	40	00	FF	15	C0	30	40	00	h @ h @ я A0@
00000010	6A	40	6	19	20	40	00	68	00	20	40	00	6A	00	FF	15	j@h @ h @ j я
00000020	E0	30	40	00	6A	00	FF	15	C4	30	40	00	00	00	00	00	a0@ j я Д0@
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Разрешение статических и внешних ссылок

Вставка ImageImportDescriptor Вставка DWORD/ASCIIIZ

Секция COFF .text

Смещение в секции COFF 00000000

Смещение в секции PE 00000000

Количество байт 00000000

Копировать

Структура COFF-ф...

Наименование	Значение
SECTIONS	
1 .text	
RELOCATION	
Address	00000001
SymIndex	0000000D
Type	DIR32
Address	00000006
SymIndex	0000000B
Type	DIR32
Address	0000000C
SymIndex	00000009
Type	DIR32
Address	00000013
SymIndex	0000000C
Type	DIR32
Address	00000018
SymIndex	0000000B
Type	DIR32
Address	00000020
SymIndex	00000007
Type	DIR32
Address	00000028
SymIndex	00000008
Type	DIR32
2 .data	
3 .drectve	

0008

Name	ExitProcess
Value	00000000
SectionNumber	SYM_UNDEFINED
Type	0000
StorageClass	EXTERNAL
NumberOfAuxSmbols	00

*** Шаг 5 - Компоновка**

- * Если все шаги сделаны правильно, то после компоновки (CTRL+F9) в каталоге проекта появится исполняемый файл, работоспособность которого необходимо проверить, запустив его на выполнение (F9).**
- * Запустить программу в дебаггере.**

*** УДАЧНОЙ ЛАБОРАТОРНОЙ!!!)**