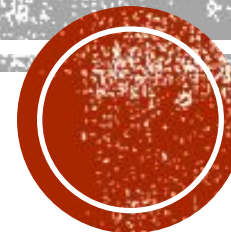


PE LINKER

Лабораторная работа №6



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				



1. НАПИСАНИЕ ПРОГРАММЫ

```

.386
.model flat,stdcall
.data
extrn GetLongPathNameA: dword
extrn MessageBoxA: dword
extrn ExitProcess: dword
.code
_start:
    push offset lpszShortPath
    push offset cchBuffer
    push offset lpszLongPath
    call  GetLongPathNameA
    push 40h
    push offset lpszShortPath
    push offset cchBuffer
    push offset lpszLongPath
    push 0
    call  MessageBoxA
    push 0
    call  ExitProcess
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!!!

адрес загрузки брать из задания!

После написания программы её необходимо откомпилировать с помощью команды:

```
ml/coff /c <имя файла>
```

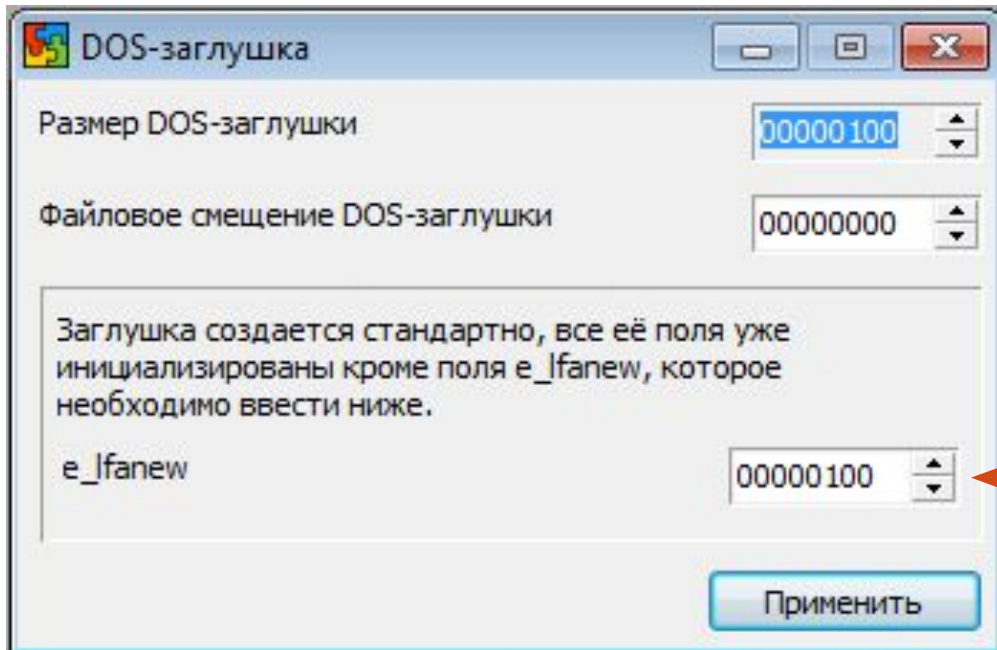


2. СОЗДАНИЕ ЗАГОЛОВКА РЕ-ФАЙЛА

- Как и в других исполняемых форматах от Microsoft, заголовков не находится 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.



2. СОЗДАНИЕ ЗАГОЛОВКА РЕ-ФАЙЛА

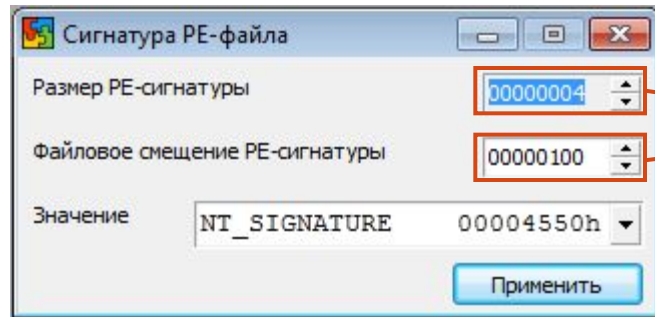


первый байт отображения файла соответствует первому байту заглушки DOS.

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



2. СОЗДАНИЕ ЗАГОЛОВКА РЕ-ФАЙЛА



+ = Файловое смещение заголовка

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



2. СОЗДАНИЕ ЗАГОЛОВКА РЕ- ФАЙЛА

Файловый заголовок

Размер файлового заголовка: 00000014

Файловое смещение заголовка: 00000104

Machine: IA-32

NumberOfSections: 0003

TimeStamp: 00000000

PointerToSymbolTable: 00000000

NumberOfSymbols: 00000000

SizeOfOptionalHeader: 00E0

Characteristics: 010F

= Файловое смещение дополнительного заголовка

NumberOfSections – кол-во секций = 3
(кода, данных, импорта)

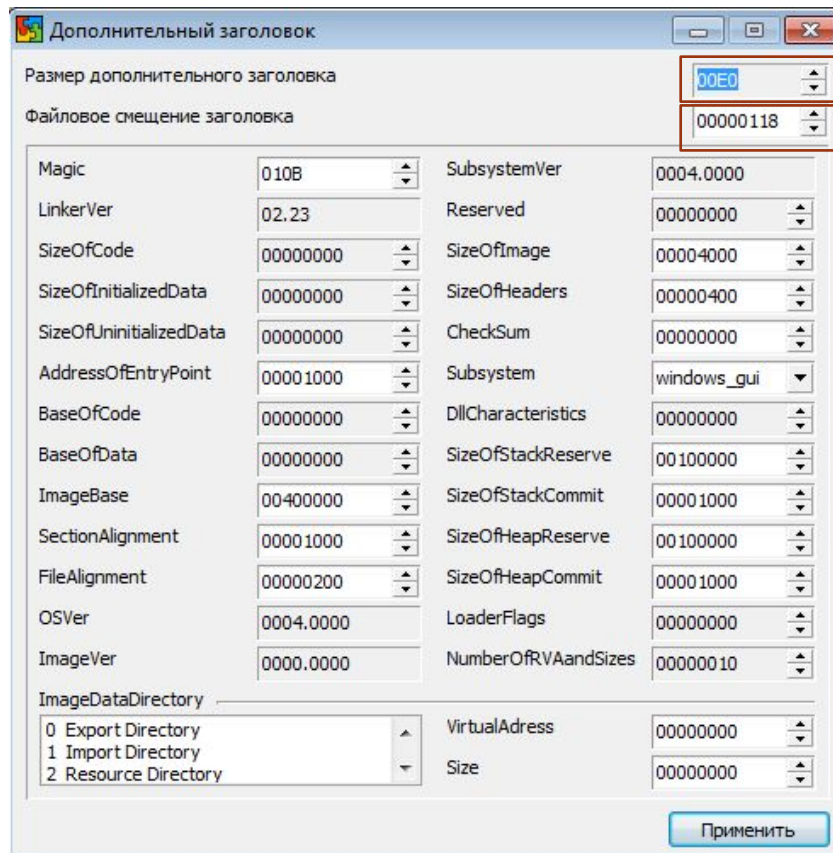
TimeStamp – время создания файла
(по-умолчанию = 0)

Characteristics

- Файл не содержит перемещений
- Файл представляет исполнимое отображение 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 отображение на съемном носителе, то скопировать и запустить из SWAP файла
- Если отображение 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)



2. СОЗДАНИЕ ЗАГОЛОВКА РЕ-ФАЙЛА



+ = Файловое смещение таблицы секций

ImageBase – адрес загрузки (см. шаг 1)

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



2. СОЗДАНИЕ ЗАГОЛОВКА РЕ- ФАЙЛА

`AddressOfEntryPoint` = 1000 (входная точка главного потока = **RVA** данных секции кода(.text))

`SectionAlignment` \geq 1000 (**const** Кратность выравнивания секций в памяти = размер страницы)

`FileAlignment` \geq 200 (**const** Кратность выравнивания секций на диске = размер сектора винчестера)

`SizeOfImage` = `VirtualAddress`(последней секции) + `VirtualSize`(последней секции) = 3000+1000=4000

`SizeOfHeaders` = 400 (**const** = размер всех заголовков и таблицы секций)



2. СОЗДАНИЕ ЗАГОЛОВКА РЕ- 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)



3. СОЗДАНИЕ СЕКЦИЙ РЕ-ФАЙЛА

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

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)



3. СОЗДАНИЕ СЕКЦИЙ РЕ-ФАЙЛА

Для секции кода(.text)

Characteristics

- Секция содержит программный код
- 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
- Секция предназначена для записи

Для секции данных(.data и .idata)

Characteristics

- Секция содержит программный код
- 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
- Секция предназначена для записи



3. СОЗДАНИЕ СЕКЦИЙ РЕ-ФАЙЛА

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	00	00	00	00	68	00	00	00	00	FF	15	00	00	00	00	h h я
00000010	6A	40	68	00	00	00	00	68	00	00	00	00	6A	00	FF	15	j@h h j я
00000020	00	00	00	00	6A	00	FF	15	00	00	00	00	00	00	00	00	j я
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	
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Вставка из секций COFF Заполнение ImageImportDescriptor Вставка DWORD/ASCIIZ

Секция COFF:

Копировать всю секцию

Смещение в секции COFF:

Смещение в секции PE:

Количество байт:



3. СОЗДАНИЕ СЕКЦИЙ РЕ- ФАЙЛА

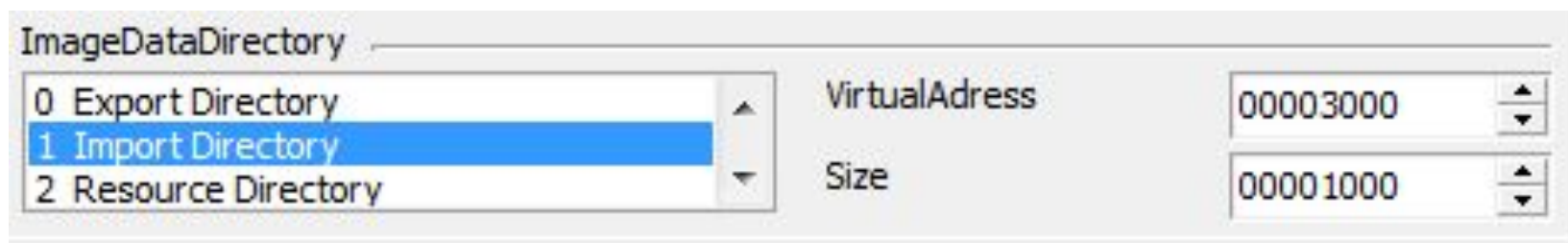
- Если это не секция “.idata” то
 1. Клик мышкой на ячейку (0;0)
 2. В нижней части всплывшего окна выбираем вкладку «Вставка из секции COFF»
 3. устанавливаем в поле «Секция COFF» открывшейся панели имя совпадающее с именем этой секции.
 4. устанавливаем в поле «Копировать всю секцию» открывшейся панели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



3. СОЗДАНИЕ СЕКЦИЙ РЕ-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.
 - Если это секция “.idata” то
 - Предварительно подключаем таблицу импорта в заголовке.



Конструктор 3-й секции «.idata»

Адрес	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDE:	
00000000	B0	30	00	00	00	00	00	00	00	00	00	00	30	60	00	00	°0	0`
00000010	C0	30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	A0	
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Вписываем массив
IMAGE_IMPORT_DESCRIPTOR
 Для каждого модуля
 Для этого надо заполнить поля во вкладке
IMAGE_IMPORT_DESCRIPTOR

000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Вставка из секций COFF | Заполнение | ImageImportDescriptor | Вставка DWORD/ASCIIZ

OriginalFirstThunk: 000030B0

Name: 00006030

TimeDateStamp: 00000000

FirstThunk: 000030C0

ForwarderChain: 00000000

Смещение в секции PE: 00000000

Вставить

Это смещение (RVA) массива двойных слов. Оно равно $VirtualAddress(секции .idata) + адрес_на_указатель_функций = 3000 + B0 = 30B0$

$VirtualAddress$ секции+смещение на строку с модулем = $3000+60$

$OriginalFirstThunk+0fh$



00000050	6B	65	72	6E	65	6C	33	32	2E	64	6C	6C	00	00	00	00	kernel32.dll
00000060	75	73	65	72	33	32	2E	64	6C	6C	00	00	00	00	00	00	user32.dll

Вписываем все подключаемые модули с помощью вставки **ASCIIZ**

00000070	00	00	47	65	74	4C	6F	6E	67	50	61	74	68	4E	61	6D	GetLongPathNam
00000080	65	41	00	00	00	00	00	00	00	00	00	00	00	00	00	00	eA
00000090	00	00	4D	65	73	73	61	67	65	42	6F	78	41	00	00	00	MessageBoxA
000000A0	00	00	45	78	69	74	50	72	6F	63	65	73	73	00	00	00	ExitProcess

Вписываем все подключаемые модули с помощью вставки **ASCIIZ**



	00	10	20	30	40	50	60	70
0		▶		0	@	P	'	p
1	☺	◀	!	1	A	Q	a	q
2	☹	↕	"	2	B	R	b	r
3	♥	!!	#	3	C	S	c	s
4	♦	¶	\$	4	D	T	d	t
5	♣	§	%	5	E	U	e	u
6	♠	=	&	6	F	V	f	v
7	•	±	'	7	G	W	g	w
8	◼	↑	(8	H	X	h	x
9	○	↓)	9	I	Y	i	y
A	◉	→	*	:	J	Z	j	z
B	♂	←	+	;	K	[k	{
C	♀	└	,	<	L	\	l	!
D	℞	↔	-	=	M]	m	}
E	♁	▲	.	>	N	^	n	~
F	✳	▼	/	?	O	_	o	△

	80	90	A0	B0	C0	D0	E0	F0
0	A	P	a	⋮	┌	└	Р	≡
1	Б	С	б	▴	┐	┑	С	±
2	В	Т	в	▾	└	┘	Т	>
3	Г	У	г		┌	└	У	<
4	Д	Ф	д	└	-	┌	Ф	┌
5	Е	Х	е	┌	+	└	Х	└
6	Ж	Ц	ж	└	└	┘	Ц	÷
7	З	Ч	з	┘	└	┘	Ч	≈
8	И	Ш	и	┌	└	┘	Ш	°
9	Й	Щ	й	└	└	┘	Щ	.
A	К	Ь	к	└	└	┘	Ь	.
B	Л	Ы	л	└	└	▀	Ы	√
C	М	Ъ	м	└	└	▀	Ъ	π
D	Н	Э	н	└	└	▀	Э	2
E	О	Ю	о	└	└	▀	Ю	●
F	П	Я	п	└	└	▀	Я	



Конструктор 3-й секции «idata »

Адрес	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000010	B0	30	00	00	00	00	00	00	00	00	00	00	30	60	00	00	°0 °`
00000020	C0	30	00	00	D0	30	00	00	00	00	00	00	00	00	00	A0 P0	
00000030	30	50	00	00	E0	30	00	00	00	00	00	00	00	00	00	0P a0	
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000050	6B	65	72	6E	65	6C	33	32	2E	64	6C	6C	00	00	00	kernel32.dll	
00000060	75	73	65	72	33	32	2E	64	6C	6C	00	00	00	00	00	user32.dll	
00000070	00	00	47	65	74	4C	6F	6E	67	50	61	74	68	4E	61	GetLongPathNam	
00000080	65	41	00	00	00	00	00	00	00	00	00	00	00	00	00	eA	
00000090	00	00	4D	65	73	73	61	67	65	42	6F	78	41	00	00	MessageBoxA	
000000A0	00	00	45	78	69	74	50	72	6F	63	65	73	73	00	00	ExitProcess	
000000B0	70	30	00	00	90	30	00	00	00	00	00	00	00	00	00	p0 50	
000000C0	70	30	00	00	90	30	00	00	00	00	00	00	00	00	00	p0 50	
000000D0	A0	30	00	00	00	00	00	00	00	00	00	00	00	00	00	0	
000000E0	A0	30	00	00	00	00	00	00	00	00	00	00	00	00	00	0	
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Вставка из секций COFF | Заполнение | ImageImportDescriptor | Вставка DWORD/ASCIIZ

Секция COFF: .text

Смещение в секции COFF: 00000000

Смещение в секции PE: 00000000

Количество байт: 00000000

Копировать всю секцию

Копировать

В строках B0,C0
делаем ссылки на
функции модуля
user32.dll

В строках D0,E0
делаем ссылки на
функции модуля
kernel32.dll



4. СТАТИЧЕСКИЕ ССЫЛКИ

- * 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.



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:

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



- 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.



Конструктор 1-й секции «text»

Адрес	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000	68	0E	20	40	00	00	00	00	00	00	00	00	00	00	00	00	
00000010	15	A3	30	40	00	00	00	00	00	00	00	00	00	00	00	00	
00000020	00	68	0F	20	40	00	00	00	00	00	00	00	00	00	00	00	
00000030	FF	15	D4	30	40	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	
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Вставка из секций COFF | Заполнение | ImageImportDescriptor | Вставка DWORD/ASCIIZ

Секция COFF: Смещение в секции COFF:

Копировать всю секцию

Смещение в секции PE:

Количество байт:

Конструктор 1-й секции «text»

Адрес	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000	68	0E	20	40	00	68	73	20	40	00	68	00	20	40	00	FF	h @ hs @ h @ я
00000010	15	A3	30	40	00	00	00	00	00	00	00	00	00	00	00	00	@ hs @
00000020	00	68	0F	20	40	00	00	00	00	00	00	00	00	00	00	00	я j0@ j
00000030	FF	15	D4	30	40	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	..._GetLongPathNameA
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...Value 00000000
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...SectionNumber SYM_UNDEFINED
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...Type 0000
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...StorageClass EXTERNAL
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...NumberOfAuxSmbols 00
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Вставка из секций COFF | Заполнение | ImageImportDescriptor | Вставка DWORD/ASCIIZ

Секция COFF: Смещение в секции COFF:

Копировать всю секцию

Смещение в секции PE:

Количество байт:

5. КОМПАНОВКА

- * 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.

