



ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно  
&& видно?



Напишите в чат, если есть проблемы!

Ставьте  если все хорошо

# Таблица импорта

PE Loader



1

PE Loader

Парсинг DOS заголовка

Парсинг PE заголовка

Копирование DOS/PE заголовков по адресу imageBase

Копирование всех секций по адресу VirtualAddress

Обработка таблицы импорта

Обработка таблицы перемещений

Выставление прав на секции

Передача управления на EntryPoint

Обработка TLS колбеков

Обработка ресурсов

Корректировка PEВ

Заглушка для GetModuleHandle

`readPe` считывает файл и возвращает адрес памяти

```
LPSTR fileName = "D:\\Languages\\fasm\\EXAMPLES\\HELLO\\HELLO_R2S.EXE";
LPVOID fileImage = readPe(fileName, fileSize);

printf("selfModule: 0x%08x\n", GetModuleHandle(NULL));

IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);
IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);

LPVOID peImage = VirtualAlloc(NULL, NtHeader->OptionalHeader.SizeOfImage, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
PIMAGE_SECTION_HEADER pSecHeader;

//copy image headers
MoveMemory(peImage, fileImage, NtHeader->OptionalHeader.SizeOfHeaders);
```

`readPe` считывает файл и возвращает адрес памяти

```
LPSTR fileName = "D:\\Languages\\fasm\\EXAMPLES\\HELLO\\HELLO_R2S.EXE";
LPVOID fileImage = readPe(fileName, fileSize);

printf("selfModule: 0x%08x\\n", GetModuleHandle(NULL));

IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);
IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);

LPVOID peImage = VirtualAlloc(NULL, NtHeader->OptionalHeader.SizeOfImage, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
PIMAGE_SECTION_HEADER pSecHeader;

//copy image headers
MoveMemory(peImage, fileImage, NtHeader->OptionalHeader.SizeOfHeaders);
```

Сразу после заголовка PE в файле располагается массив заголовков секций

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
        DWORD    PhysicalAddress;
        DWORD    VirtualSize;
    } Misc;
    DWORD    VirtualAddress;
    DWORD    SizeOfRawData;
    DWORD    PointerToRawData;
    DWORD    PointerToRelocations;
    DWORD    PointerToLinenumbers;
    WORD     NumberOfRelocations;
    WORD     NumberOfLinenumbers;
    DWORD    Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;

#define IMAGE_SIZEOF_SECTION_HEADER    40
```

Не указанные поля равны 0

Название	Описание
Misc.VirtualSize	Размер секции в памяти. Если это значение больше SizeOfRawData, то секция дополняется в памяти нулевыми байтами
<b>VirtualAddress</b>	<b>RVA</b> секции в памяти
SizeOfRawData	Размер секции в <b>файле</b> . Всегда кратен FileAlignment imageOptionalHeader
PointerToRawData	Смещение в <b>файле</b> до начала данных секций. Всегда кратно FileAlignment из imageOptionalHeader
Characteristics	Атрибуты секции

Перевод виртуального адреса PE файла в его файловое смещение

```
DWORD RVA2RAW(LPVOID fileImage, DWORD dwRVA)
{
    DWORD dwRawRVAAdr(0);
    IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);
    IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);
    IMAGE_SECTION_HEADER pSections = IMAGE_FIRST_SECTION(NtHeader);

    if (!pSections)
    {
        return dwRawRVAAdr;
    }

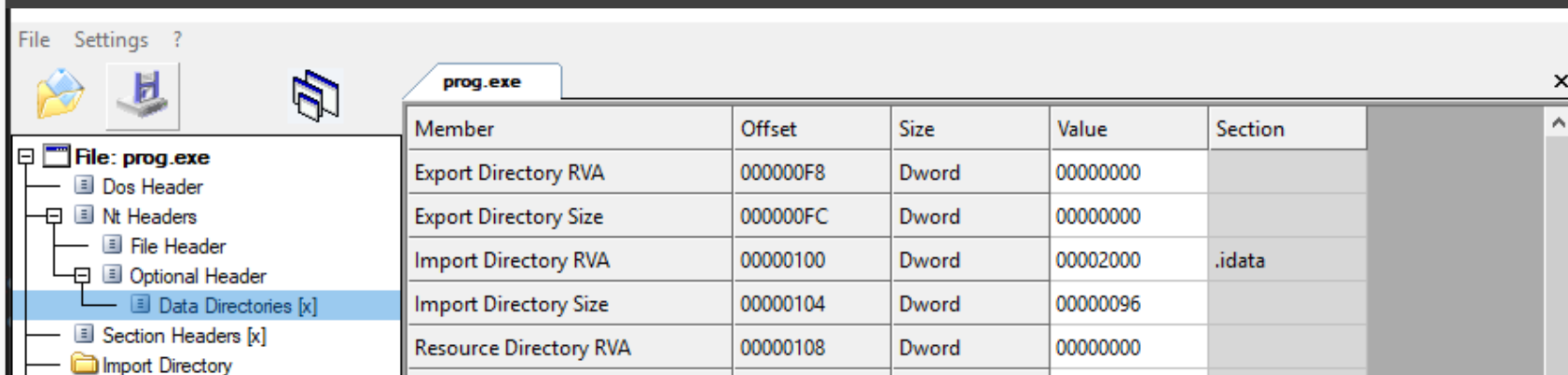
    while (pSections->VirtualAddress != 0)
    {
        if (dwRVA >= pSections->VirtualAddress &&
            dwRVA < pSections->VirtualAddress + pSections->SizeOfRawData)
        {
            dwRawRVAAdr = (dwRVA - pSections->VirtualAddress) + pSections->PointerToRawData;
            break;
        }
        pSections++;
    }

    return dwRawRVAAdr;
}
```

2

Таблица импорта

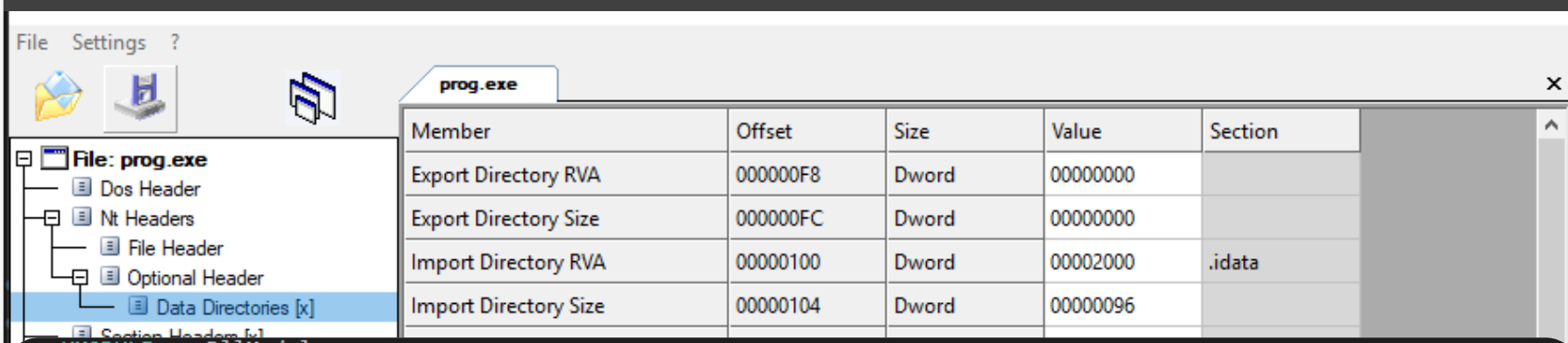
Смещение до import Table получается из массива  
DataDirectory - по индексу == 1



The screenshot shows a software interface for parsing a PE file named 'prog.exe'. On the left, a tree view shows the file structure with 'Data Directories [x]' selected. On the right, a table displays the details of the data directories.

Member	Offset	Size	Value	Section
Export Directory RVA	000000F8	Dword	00000000	
Export Directory Size	000000FC	Dword	00000000	
Import Directory RVA	00000100	Dword	00002000	.idata
Import Directory Size	00000104	Dword	00000096	
Resource Directory RVA	00000108	Dword	00000000	

Смещение до import Table получается из массива DataDirectory - по индексу == 1



Member	Offset	Size	Value	Section
Export Directory RVA	000000F8	Dword	00000000	
Export Directory Size	000000FC	Dword	00000000	
Import Directory RVA	00000100	Dword	00002000	.idata
Import Directory Size	00000104	Dword	00000096	

```
HMODULE curDllModule;  
FUNC_ITEM func;  
FUNCS_ARRAY result;  
int imageDescrIndex = 0, thunkIndex = 0;  
  
IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);  
IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);  
  
DWORD imageImportDescrVA = NtHeader->OptionalHeader.DataDirectory[1].VirtualAddress;  
DWORD imageImportDescrRAW = RVA2RAW(fileImage, imageImportDescrVA);  
  
PIMAGE_IMPORT_DESCRIPTOR fileImageImportDescr = (PIMAGE_IMPORT_DESCRIPTOR)((DWORD)fileImage + imageImportDescrRAW);  
vector<string> dllList;
```

По полученному смещению находится массив структур  
типа `IMAGE_IMPORT_DESCRIPTOR`

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD Characteristics;           // 0 for terminating null import descriptor
        DWORD OriginalFirstThunk;       // RVA to original unbound IAT (PIMAGE_THUNK_DATA)
    } DUMMYUNIONNAME;
    DWORD TimeDateStamp;                // 0 if not bound,
                                        // -1 if bound, and real date\time stamp
                                        // in IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT (new BIND)
                                        // O.W. date/time stamp of DLL bound to (Old BIND)

    DWORD ForwarderChain;               // -1 if no forwarders
    DWORD Name;
    DWORD FirstThunk;                  // RVA to IAT (if bound this IAT has actual addresses)
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;
```

```
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;
} IMAGE_IMPORT_DESCRIPTOR;
    DWORD OriginalFirstThunk;           // RVA to IAT (if bound this IAT has actual addresses)
    DWORD Name;
```

Каждой динамической библиотеке соответствует по одной такой структуре

Название	Описание
Characteristics	RVA адрес массива, содержащего RVA адреса имён импортируемых функций (может быть 0)
TimeDateStamp	Дата/время
ForwarderChain	Обычное значение – 0xFFFFFFFF
Name	RVA строки имени динамической библиотеки
FirstThunk	Вторая копия Characteristics

Characteristics и FirstThunk указывают на массив из структур IMAGE\_THUNK\_DATA32

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;    // PBYTE
        DWORD Function;          // PDWORD
        DWORD Ordinal;
        DWORD AddressOfData;      // PIMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA32;
typedef IMAGE_THUNK_DATA32 * PIMAGE_THUNK_DATA32;
```

Если старшее слово поля == 0x8000, то младшее слово содержит ordinal

AddressOfData + 2 – адрес имени импортируемой функции

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;    // PBYTE
        DWORD Function;          // PDWORD
        DWORD Ordinal;
        DWORD AddressOfData;     // PIMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA32;
typedef IMAGE_THUNK_DATA32 * PIMAGE_THUNK_DATA32;
```

Если старшее слово поля == 0x8000, то младшее слово содержит ординал

# Пример: Получить имя импортируемой функции в hiew

1. Переходим на таблицу импорта: F8->F10, выбираем Import

Name	RVA	Size
Export	00000000	00000000
<b>Import</b>	<b>00002000</b>	<b>000000B0</b>
Resource	00000000	00000000
Exception	00000000	00000000
Security	00000000	00000000
Fixups	00004000	0000001A
Debug	00000000	00000000
Description	00000000	00000000
MIPS GP	00000000	00000000
TLS	00000000	00000000
Load config	00000000	00000000
Bound Import	00000000	00000000
Import Table	00000000	00000000
Delay Import	00000000	00000000
COM Runtime	00000000	00000000
(reserved)	00000000	00000000

# Пример: Получить имя импортируемой функции в hiew

2. Первые 4 байта (поле Characteristics - 0x00002058) соответствуют первому элементу массива структур IMAGE\_IMPORT\_DESCRIPTOR. Это RVA структуры IMAGE\_THUNK\_DATA32.

```
.2058| 00 00 00 00-00 00 00 00-00 00 00 00
.004011D0: 00 00 00 00-00 00 00 00-00 00 00 00
.004011E0: 00 00 00 00-00 00 00 00-00 00 00 00
.004011F0: 00 00 00 00-00 00 00 00-00 00 00 00
.00402000: 58 20 00 00-00 00 00 00-3C 20 00 00 X <
.00402010: 64 20 00 00-90 20 00 00-00 00 00 00 d P
.00402020: 4A 20 00 00-98 20 00 00-00 00 00 00 J Ш
.00402030: 00 00 00 00-00 00 00 00-4B 45 52 4E KERN
.00403030: 00 00 00 00-00 00 00 00-4B 42 23 4E KERN
.00403030: 4A 20 00 00-08 20 00 00-00 00 00 00 J Ш
```

Перейдём по адресу: F5, вводим 0x2058

# Пример: Получить имя импортируемой функции в hiew

3. Получаем поле AddressOfData структуры IMAGE\_THUNK\_DATA32 = 0x00002070

По адресу 0x00002070 + 2 должно быть имя первой импортируемой функции

```
00402050: 2E 44 4C 4C-00 00 00 00-70 20 00 00-7E 20 00 00 .DLL p ~
00402060: 00 00 00 00-70 20 00 00-7E 20 00 00-00 00 00 00 p ~
00402070: 00 00 45 78-69 74 50 72-6F 63 65 73-73 00 00 00 ExitProcess
00402080: 47 65 74 43-6F 6D 6D 61-6E 64 4C 69-6E 65 41 00 GetCommandLineA
00402090: A0 20 00 00-00 00 00 00-A0 20 00 00-00 00 00 00 a a
004020A0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00 " "
```

Перейдём по адресу: F5, вводим 0x2072

# Пример: Получить имя импортируемой функции в hiew

3. Получаем поле AddressOfData структуры IMAGE\_THUNK\_DATA32 = 0x00002070

По адресу 0x00002070 + 2 должно быть имя первой импортируемой функции

```
.00402050: 2E 44 4C 4C-00 00 00 00-70 20 00 00-7E 20 00 00 .DLL p ~
.00402060: 00 00 00 00-70 20 00 00-7E 20 00 00-00 00 00 00
.00402070: 00 00 45 78-69 74 50 72-6F 63 65 73-73 00 00 00 ExitProcess
.00402080: 47 65 74 43-6F 6D 6D 61-6E 64 4C 69-6E 65 41 00 GetCommandLineA
.00402090: A0 20 00 00-00 00 00 00-A0 20 00 00-00 00 00 00 a a
.004020A0: 00 00 4D 65-73 73 61 67-65 42 6F 78-41 00 00 00 MessageBoxA
.004020B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.004020C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.004020D0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.004020E0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Перейдём по адресу: F5, вводим 0x2072

Поле `AddressOfData` массива `FirstThunk` заменяется виртуальными адресами импортируемых функций загрузчиком PE файлов!

Массив `Characteristics` остаётся без изменения

```
.00402050: 2E 44 4C 4C-00 00 00 00-70 20 00 00-7E 20 00 00  .DLL    p    ~
.00402060: 00 00 00 00-70 20 00 00-7E 20 00 00-00 00 00 00
.00402070: 00 00 45 78-69 74 50 72-6F 63 65 73-73 00 00 00  ExitProcess
.00402080: 47 65 74 43-6F 6D 6D 61-6E 64 4C 69-6E 65 41 00  GetCommandLineA
.00402090: A0 20 00 00-00 00 00 00-A0 20 00 00-00 00 00 00  a      a
.004020A0: 00 00 4D 65-73 73 61 67-65 42 6F 78-41 00 00 00  MessageBoxA
.004020B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.004020C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Перейдём по адресу: F5, вводим 0x2072

Вопросы???





Пакулов Артур

[A.Pakulov.Otus@Gmail.com](mailto:A.Pakulov.Otus@Gmail.com)

Спасибо  
за внимание!

