Шестая межрегиональная научная конференция
«Шаг в будущее, Москва»

регистрационный номер

Информатика и системы управления

Система перехвата вызовов функций Win32 API

Автор: Попов Артём Михайлович
лицей 1580, 11 класс

Руководитель: *Дединский Илья Рудольфович*
преподаватель программирования лицея 1580

Москва 2003

Аннотация

регистрационный номер

К работе **«Система перехвата вызовов функций Win32 API»**

Автор: ***Попов Артём Михайлович***
лицей 1580, 11 класс

Научный руководитель: ***Дединский Илья Рудольфович***
преподаватель программирования лицея 1580

В работе были изучены методы перехвата API-функций и в результате проведённых исследований был разработан оптимальный метод перехвата и разработан программный комплекс для перехвата API-функций и автоматизированного создания библиотек перехвата. Были созданы библиотеки перехвата для основных системных модулей операционной системы Windows. Имеются примеры применения системы перехвата для стандартных пользовательских и системных программ ОС Windows.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Win32 API – множество функций, предназначенных для взаимодействия прикладных программ с операционной системой Windows. Большинство программ, работающих в этой операционной системе, используют те или иные функции Win32 API (далее для краткости будем называть их просто API-функциями). Они могут использоваться для работы с оконным интерфейсом, с файловой системой, с сетевыми сервисами и т. д.. Вместе с тем, работа с API-функциями и отладка программ, их использующих, представляет определённые сложности. Они связаны с высокой частотой вызовов API-функций, а также с трудностью отслеживания значений многочисленных параметров, передаваемых функциям. В связи с этим в ходе работы с программой появляется необходимость перехвата вызовов API-функций. К сожалению, перехват API-функций также может использоваться и для взлома программ. Однако любой метод взлома также, с другой стороны, является и средством тестирования защищённости программы.

Программы для перехвата API-функций являются удобным средством отладки, однако они не очень распространены и обладают ограничениями функциональных возможностей. Это и явилось основанием для разработки настоящего программного комплекса.

1. Цель и задачи работы

Целью настоящей работы являлась разработка программного комплекса для перехвата API-функций и автоматизированного создания библиотек перехвата.

Задачи работы:

1) Изучение методов перехвата вызовов API-функций;
2) Выбор подходящего метода перехвата;
3) Написание вспомогательных программ для генерирования функций перехвата;
4) Создание библиотек перехвата для основных системных модулей.
5) Создание средства отображения и сохранения информации о перехваченных функциях;

2. Сравнительный анализ существующих разработок

Название: APIMon (см. рис. 1)

Разработчик: Microsoft (http://www.microsoft.com/)

Программа написана разработчиком самой операционной системы, поэтому вполне можно ожидать от неё наиболее высоких результатов, но в ходе проведённых экспериментов она не выдала корректных результатов ни в ОС Windows 2000, ни в ОС Windows 98 (по крайней мере не удалось восстановить условия, необходимые для правильной работы программы), но зато в ней был обнаружен встроенный отладчик, который обычно отсутствует в подобных программах.

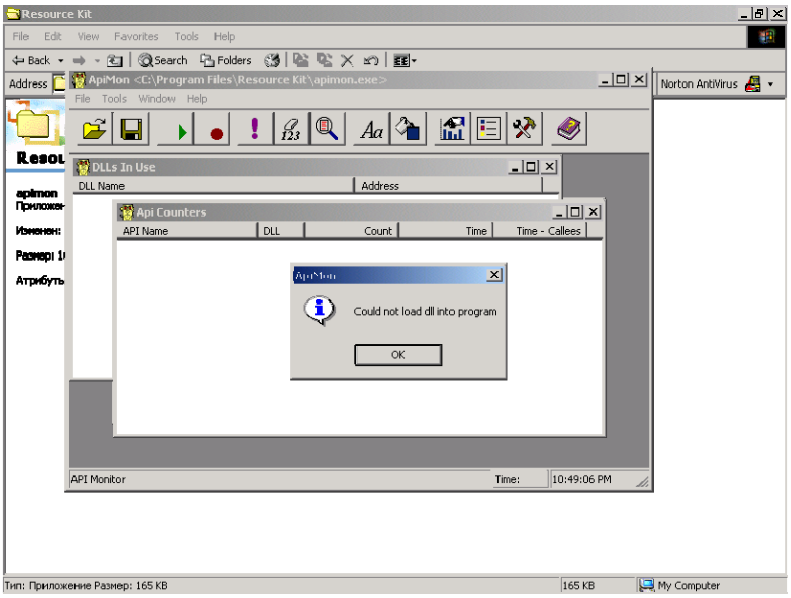

Рис. 1. Программа ApiMon

Название: APIMonitor (см. рис 2)

Разработчик: Rohitab Batra (www.rohitab.com/apimonitor).

Программа перехватывает вызовы функций всеми процессами в системе путём инсталляции своего драйвера и входа в режим отладки процессов (это лишь предположение, так как исходные тексты программы автором не публикуются). Программа является прекрасным образцом написания подобных утилит, т. к. она выполнена очень качественно, удобна в управлении, имеет удобный интерфейс управления. К недостаткам программы относится то, что при работе с большим количеством перехватываемых функций, программа сильно замедляет работу всей операционной системы.

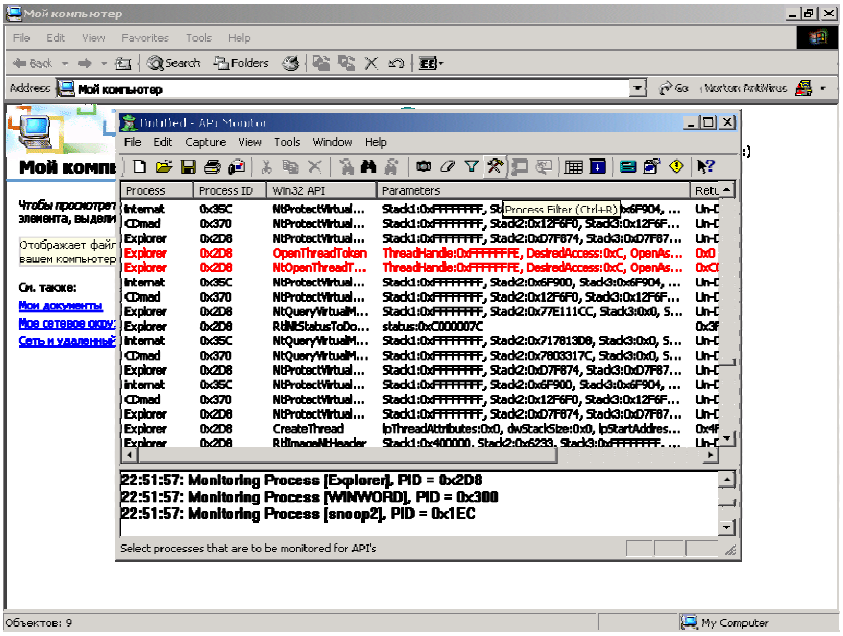

Рис. 2 Программа ApiMonitor

Название: APISpy (APIS32)

Разработчик: Vitaly Evseenko (APIS@Biosys.net)

Эта программа имеет удобный интерфейс пользователя и обладает гибкостью настройки. Алгоритм работы программы автором не публиковался и, к сожалению, остался невыясненным в ходе работы. Единственный минус этой программы — она перехватывает не все функции.

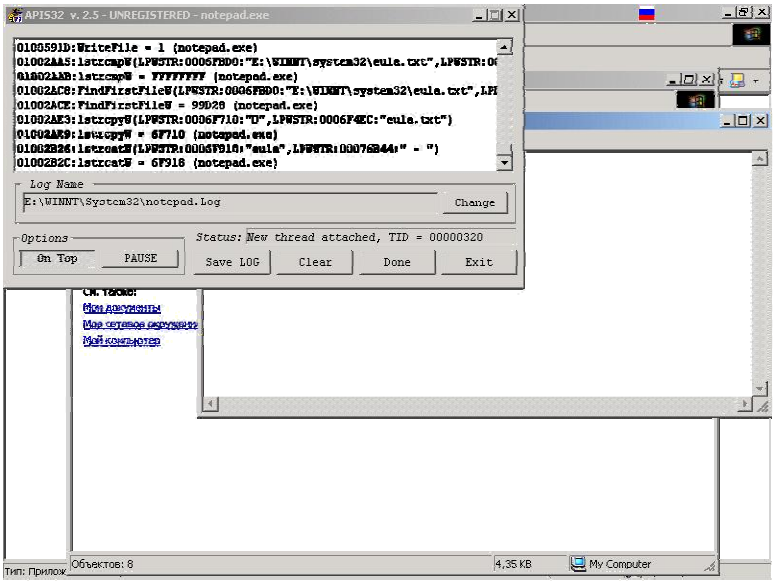

Рис. 3. Программа APISpy (APIS32)

В табл. 1 приведены сравнительные характеристики описанных выше разработок.

Табл. 1. Сравнительные характеристики существующих разработок.

Название	**Достоинства**	**Недостатки**
APIMon	Наличие встроенного отладчика.	В ходе проведённых экспериментов программа не выдала корректных результатов (по крайней мере, восстановить условия, необходимые для её работы, не удалось).
APIMonitor	Удобный интерфейс пользователя, перехват большинства API-функций (в том числе и недокументированных).	Для установки программы требуются права администратора в операционной системе.
APISpy (APIS32)	Высокая скорость работы, удобство использования.	Перехват малого количества функций, программа не отображает имена параметров функций.

3. Математическое обеспечение

Так как все методы перехвата так или иначе основаны на особенностях строения исполняемого файла, то стоит рассмотреть структуру стандартного Win32 EXE файла.

3.1. Структура стандартного Win32 EXE-файла

Стандартный для Windows формат исполняемого файла называется PE-форматом (из-за сигнатуры PE (Portable Executable), обязательной для этих файлов). На рис. 4 представлен формат такого файла. Все составные части PE файла для данной задачи не нужны (их подробное описание можно найти в книге В. Юрова “Assembler. Специальный справочник”). Нас интересует секция импортируемых функций .idata, чаще всего она называется таблицей импорта. Рассмотрим подробнее таблицу импорта, т. к. именно особенности импорта функций позволяют перехватывать их вызовы.

0000:0000 0000:003Ch	MZ “Заглушка” (stub)MS-DOS (“This program requires Microsoft Windows”) xxxx
0000:xxxx	PE\0\0
	Основной PE-заголовок (20 байт)
	Доп. PE-заголовок (переменная длина)
	Таблица секций
	Секция кода .text
	Секция данных .data
	Секция данных .bss
	Секция экспортируемых функций .edata
	Секция импортируемых функций .idata
	Другие секции

Рис. 4 Структура стандартного Win32 EXE файла

3.2. Таблица импорта

С точки зрения структуры таблица импорта состоит из трёх частей:

1) Массива с описанием используемых DLL-библиотек - ArrayDllDef;
2) Двух массивов с адресами импортируемых функций – ArrayAddressImpFunc;
3) Массива с описанием имён импортируемых функций – ArrayNameImpFunc;

Эти четыре массива располагаются в секции .idata последовательно друг за другом. Массив с описанием используемых библиотек ArrayDllDef предназначен для указания местоположения в секции .idata имён dll-библиотек и массивов с адресами импортируемых функций ArrayAddressImpFunc для каждой из используемых dll-библиотек. Структура массива ArrayDllDef показана в табл. 1.

Таблица 2. Элемент массива ArrayDllDef

Смещение	Размер, байт	Назначение
+00h	4	Значение смещения от начала секции .idata к началу массива с адресами импортируемых функций ArrayAddressImpFunc
+04h	4	Зарезервировано (0)
+08h	4	Зарезервировано (0)
+0Ch	4	Значение смещения от начала секции .idata до начала строки с именем dll-библиотеки
+10h	4	Значение смещения от начала секции .idata к началу массива с адресами импортируемых функций ArrayAddressImpFunc (вторая копия

Количество элементов этого массива равно количеству библиотек, функции которых импортируются данной программой. Массив кончается элементом с нулевыми полями.

Массив ArrayAddressImpFunc следует за массивом ArrayDllDef. Этот массив состоит из двойных слов, представляющих собой смещения относительно начала секции .idata к элементу массива ArrayNameImpFunc. Структура элемента массива ArrayNameImpFunc рассматривается в таблице 2.

Таблица 3. Элементы массива ArrayNameImpFunc

Смещение	Размер,байт	Назначение
+00h	2	Номер экспорта из dll-библиотеки для данной импортируемой функции
+02h	переменный	Строка с именем импортируемой функции

Содержимое элемента массива ArrayAddressImpFunc зависит от того, в какой момент времени оно рассматривается и в какой копии массива находится. Ответ на данный вопрос следует из особенностей механизма загрузки EXE-файла. Рассмотрим его.

3.3. Механизм загрузки Win32 EXE файла

Пока PE файл находится на диске, содержимое двух копий массива с адресами импортируемых функций ArrayAddressImpFunc абсолютно идентично. Каждый элемент массива ArrayAddressImpFunc представляет собой смещение относительно начала секции .idata к элементу массива ArrayNameImpFunc. Каждый элемент массива ArrayNameImpFunc представляет собой структуру, содержащую номер экспорта функции из соответствующих библиотек и имя функции. Номером экспорта называется специальное число, являющееся уникальным для каждой экспортируемой функции в библиотеке. Фактически это индекс в массиве адресов экспортируемых библиотекой функций. Массивы ArrayAddressImpFunc в первой копии не изменяются в процессе загрузки. Массивы во второй копии изменяются следующим образом. Загрузчик просматривает массив ArrayAddressImpFunc, по содержащимся в нём смещениям находит адреса (или импортные номера) функций в соответствующих dll-библиотеках и замещает содержимое двойных слов на истинные значения адресов функций в памяти.

Таким образом, такая схема формирования адресов импортируемых функций позволяет перехватить функции любой dll-библиотеки. Достаточно лишь заменить имя модуля, и загрузчик будет искать функции не в стандартной библиотеке, а в библиотеке, имя которой было подставлено вместо стандартного. Таким образом идея механизма перехвата понятна, и можно перейти к её реализации.

4. Описание программного комплекса

4.1. Состав программного комплекса

Рассмотрим состав программного комплекса.

1) Generator2 – генератор функций перехвата
2) GetProto – программа, извлекающая из заголовочных файлов прототипы функций
3) TD2Def – разборщик типов
4) DLLs\User32 – модуль перехвата для user32.dll
5) DLLs\Kernel32\Kernel32 – модуль перехвата для kernel32.dll
6) DLLs\Kernel32\GDI32 – модуль перехвата для gdi32.dll

Напомним, что библиотеки user32.dll, kernel32.dll, gdi32.dll являются системными библиотеками. В табл. 2 приведено функциональное назначение этих библиотек.

Табл. 4. Функциональное назначение системных библиотек

Имя	**Функциональное назначение**
kernel32.dll	Обеспечение взаимодействия прикладных программ с ядром операционной системы Windows. В ней содержатся функции, предназначенные для работы с файловой системой, с процессами, с языковыми настройками операционной системы Windows и т. д..
user32.dll	Работа с интерфейсом пользователя, с языковыми настройками операционной системы Windows, с ресурсами и т. д..
gdi32.dll	Обеспечение работы прикладных программ с графическими устройствами (с монитором, принтерами и т.п.).

4.2. Разработка методики перехвата API-функций

Из разделов 4.1-4.3 ясно, что изменяя содержимое таблицы импорта, можно сделать так, что вызываться будет не стандартная функция, а специальная функция перехвата. Это делается простым изменением имени модуля, соответствующего перехватываемой функции, на имя модуля, в котором содержится функция перехвата. Но если изменять только имя модуля, то имена функций останутся такими же, какими они были в стандартной библиотеке. Следовательно, функция перехвата должна иметь имя, совпадающее с именем стандартной функции. Далее для нормальной работы программы из функции перехвата необходимо передавать управление стандартной функции. Однако в таком случае имя пере-

хватывающей функции будет совпадать с именем стандартной функции, и, соответственно, передать управление API функции традиционным способом (по имени) будет нельзя. Из данной ситуации существует два выхода:

1) Взятие адреса функции с помощью функций LoadLibrary и GetProcAddress, а затем переход по этому адресу.

2) Написание библиотеки переходника (двухуровневый перехват), состоящей из функций вида:

```
int MyMessageBoxA (HWND hWnd, LPSTR lpMessage, LPSTR Caption, UINT uType)
{
   return MessageBoxA (hWnd, lpMessage, lpCaption, uType);
}
```

Т. е. функция перехвата вызывает соответствующую функцию из библиотеки переходника, а та, в свою очередь, передаёт управление стандартной функции.

В описываемой программе для перехвата используется первый способ, т. к. он удобнее и быстрее. Однако в функциях перехвата для библиотеки KERNEL32.DLL используется второй способ для вызова функций GetProcAddress, LoadLibrary и IsBadCodePtr, т. к. эти функции содержатся в самом модуле перехвата.
Таким образом, общая схема перехвата такова (см. рис. 2):

1) Загружается основной загрузчик, на входе у которого имя обрабатываемого EXE файла. Он изменяет таблицу импорта EXE файла так, что имена стандартных модулей в таблице импорта меняются на имена модулей перехвата: USER32 –> _SER32, KERNEL32 –> _ERNEL32, GDI32 –> _DI32.DLL, где _SER32, _ERNEL32 и _DI32 — модули перехвата для соответствующих системных библиотек.

2) Запускается EXE файл и загружает выше перечисленные модули перехвата.

3) Программа начинает свою работу, вызывает функции API, но не из стандартного модуля, а из модуля перехвата перехвата. Функции перехвата заносят информацию о перехваченных функциях в файл spy.txt.

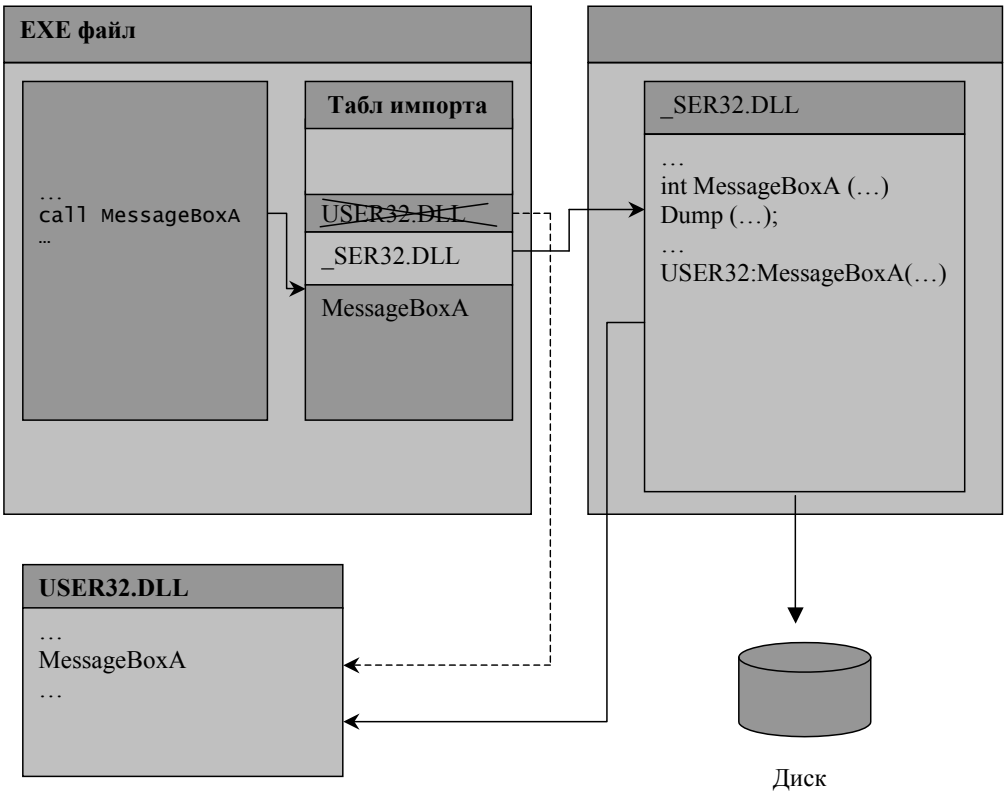

Рис. 5. Схема работы перехватчика API-функций.

4.2. Стандартная функция перехвата

Принцип работы программы был описан в разделе (4). Рассмотрим детально типичную функцию перехвата на примере перехватчика для функции CreateWindowStation из модуля USER32.DLL (см. листинг 1). Напомним, что она вызывается вместо стандартной функции. Её задача – сохранить информацию о перехваченной функции (имя, параметры, возвращённое значение) на диск и передать управление стандартной функции.

```
1. extern "C" __declspec(dllexport) void* __stdcall CreateWindowStationW(
                                        char* lpwinsta,
                                        unsigned long dwReserved,
                                        unsigned long dwDesiredAccess,
                                        void* lpsa)
2. {
3. typedef void* __stdcall ptr (char*,unsigned long,unsigned long,void*);
4. static int FindStrCalled = 0,
5.            StrFound      = 0;
6. if ( !FindStrCalled )
7. {
8.       StrFound = StrFind ( "CreateWindowStationW" );
9.       FindStrCalled = 1;
10.      }
11.      if ( StrFound )
12.      {
13.            dump ("USER32 : CreateWindowStationW (lpwinsta :
                                                (%p)\"%s\";"
                                    “dwReserved : %lu;”
                                    “dwDesiredAccess : %lu;”
                                    “lpsa : %p;);\n",
                                    lpwinsta,
```

```
                                                (IsBadCodePtr (lpwinsta) ?
                                                "*** BAD PTR ***" :
                                                lpwinsta ),
                                                dwReserved,
                                                dwDesiredAccess,
                                                lpsa);
14.       }
15.       int  __hInst = LoadLibrary ( "USER32.DLL" );
16.       int    proc  = GetProcAddress ( __hInst, "CreateWindowStationW" );
17.       ptr*   stdfn = (ptr*)proc;
18.       void* res = stdfn (lpwinsta,dwReserved,dwDesiredAccess,lpsa);
19.       return res;
20.       }
```

Листинг 1. Функция-перехватчик

Строка (1): заголовок функции; *extern "C"* — это ключевое слово позволяет экспортировать функции не по изменённому имени, а по стандартному. Это сделано для того, чтобы модуль перехвата экспортировал функции так же, как и стандартные системные модули.

Строка (3): Объявление типа указателя на данную функцию. Это понадобится на последнем этапе работы функции перехвата – возврате управления стандартной API функции.

Строки (4)-(12): эти строки обеспечивают настраиваемость работы перехвата. Функция *StrFind* ищет в файле fns.dat, созданном загрузчиком или пользователем, имя перехватываемой функции. Если оно найдено, *StrFind* возвращает 1 (иначе 0). Переменная *StrFindCalled* равна 1, если функция *StrFind* уже вызывалась (иначе 0). В переменную *StrFound* кэшируется результат вызова функции *StrFind.* Таким образом, если имени функции в файле fns.dat нет (или нет самого файла), то функция дампиться не будет, а последовательность условных переходов и статические переменные нужны для того, чтобы функция обращалась к файлу fns.dat на диске только один раз, из-за этого скорость работы перехвата увеличивается.

Строка (13): Вызов функции Dump, которая сохраняет информацию о вызванной функции на диск.

Строки (16)-(17): Нахождение адреса стандартной API функции. Это нужно для будущей передачи управления на неё.

Строка (18): объявление указателя на стандартную API функцию и приведение его к нужному для корректного вызова типу.

Строка (19): Передача управления на стандартную функцию. При этом возвращаемое значение кладётся в переменную rez.

Строка (20): возврат управления в программу.

Конечно же, вручную написать такую функцию для каждой стандартной API функции невозможно, поэтому был написан генератор функций перехвата. Рассмотрим его.

4.3. Генератор функций перехвата

Генератор функций перехвата – это программа, которая по имеющемуся списку прототипов функций строит функцию перехвата для каждого из них. Структура всех функций перехвата одинакова и описана в разделе 5.2. Генератор состоит из трёх модулей.

Первый модуль – это модуль, содержащий функцию GetFn и вспомогательные функции к ней. Функция GetFn раскладывает прототип функции на имя функции, тип функции и её параметры. Параметры в свою очередь раскладываются на имена и типы. Вся полученная информация кладётся в структуру FN. Грамматический разбор прототипа осуществляется методом рекурсивного спуска.

Второй модуль – модуль, содержащий функцию GetTemplate. Данная функция ставит в соответствие базовому типу подстановочную строку для функции семейства print. Например, переменной “int a” она ставит в соответствие подстановочную строку “%d” и набор параметров “a”, переменной “int* lpa” она ставит в соответствие подстановочную строку “(0x%p) %d” и набор параметров “lpa, *lpa”.

Третий модуль – модуль, генерирующий функции перехвата по прототипу, используя первые два модуля.

Исходный код генератора находится в папке src\gen2

- Генератор: gen2_2.cpp
- Первый модуль (GetFn): getfn.h
- Второй модуль (GetTemplate): template.h

Рассмотрим пример создания функции перехвата.

4.4. Пример создания функции перехвата

Пусть дан прототип: void* Func(int a,char* b,int* c,void* d).

1) Первой работает функция GetFn. Она вернёт структуру Fn следующего содержания:

   Fn -> name = “Func”;
   Fn -> type -> name = “void*”;
   Fn -> varsc = 4;

   Fn -> vars [0] -> name = “a”;
   Fn -> vars [0] -> type -> name = “int”;

   Fn -> vars [1] -> name = “b”;
   Fn -> vars [1] -> type -> name = “char*”;

   Fn -> vars [2] -> name = “c”;

   Fn -> vars [2] -> type -> name = “int*”;

   Fn -> vars [3] -> name = “d”;
   Fn -> vars [3] -> type -> name = “void*”;

2) Далее в выходной файл записывается:
   “extern “C” __declspec(dllexport) <имя_типа_ функции > __stdcall <имя_функции>(
   где имя типа и имя функции берутся из структуры Fn.

3) Затем в цикле до Fn -> varsc в выходной файл записываются параметры, имена и типы которых берутся из структуры Fn.

4) Далее генерируется следующая строка:
   “typedef <имя_типа_функции> __stdcall ptr (<типы_параметров_в_цикле)”

5) Далее с помощью функции GetTemplate определяются строки форматирования и параметры для функции dump. Рассмотрим их для каждого параметра:
   a) int a -> “%d”, “a”
   b) char* b -> “(0x%p) \”%s\””, “IsBadCodePtr (b) ? \“*** BAD PTR ***\” : b, b”

      Здесь указатель b проверяется на валидность, т. е. проверяется, имеется ли у доступ к памяти по этому указателю или нет. Это необходимо для автоматического генерирования функций перехвата для API-функций, работающих с ресур-

сами. Дело в том, что большинству из этих функций передаётся либо имя ресурса в виде указателя на строку, либо его идентификатор (число, уникальное для каждого ресурса в программе). Но идентификатор передаётся в том же параметре, что и указатель. И если по указателю получить доступ к данным можно, то при попытке использования идентификатора ресурса как указателя, произойдёт неисправимая ошибка (access violation). Проверка указателя на валидность помогает избежать этого.

c) int* c -> “(0x%p) %d”, “c, *c”
d) void* d -> “0x%p”, “d”
e) С помощью функций strcat и strcpy формируются параметры, которые будут передаваться в функцию dump из функции перехвата (генерируется строка вида строки (13) из листинга 1).

6) На основе полученных данных генерируется тело функции (строки (4)-(17) в листинге 1 разделе 5.2)

7) Генерируется строка
“<имя_типа_функции> rez = (ptr*)stdfn (<имена_параметров_ функции _в_цикле>);”

8) Если функция типа “void”, то записывается закрывающая фигурная скобка, иначе “return rez; }”

4.5. Замечания по поводу перехвата функций GetProcAddress, LoadLibrary и IsBadCodePtr

Три выше перечисленные функции используются самими функциями перехвата, поэтому при создании библиотеки перехвата для KERNEL32.DLL я использовал библиотеку-переходник. Принцип работы с библиотекой-переходником таков (см. рис. 6): функция перехвата вызывает не GetProcAddress (или др.), а _GetProcAddress из библиотеки переходника, а _GetProcAddress в свою очередь вызывает стандартную функцию GetProcAddress.

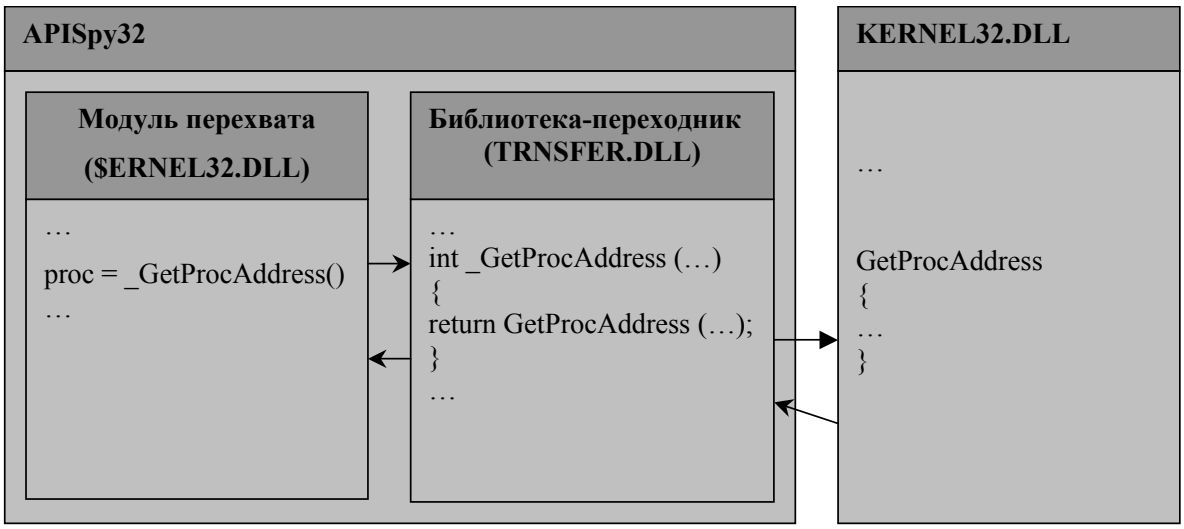

Рис. 6. Схема работы библиотеки-переходника.

Пример функции из библиотеки-переходника (см. рис.6):

```
extern “C” __declspec(dllexport) int _GetProcAddress (void* hInstance, char* FnName)
        {
                return GetProcAddress (hInstance, FnName);
        }
```

5. Методическое обеспечение

5.1. Системные требования

Минимальные аппаратные требования:

1. Процессор Pentium 133 MHz
2. Оперативная память 32 MB
3. Видеокарта с 2 Mb памяти
4. Microsoft Compatible Mouse

Рекомендованная конфигурация:

1. Процессор Pentium 200 MHz
2. Оперативная память 64 MB
3. Видеокарта с 4 Mb памяти
4. Microsoft Compatible Mouse

Программные требования:

1. Установленная ОС Windows NT/2000.

5.2. Руководство пользователя

5.2.1. Использование вспомогательных программ

1) Программа GETPROTO.

   Эта программа извлекает из заголовочных файлов прототипы функций. Каждый прототип должен начинаться со специального идентификатора (далее будем называть его базовым типом). Это условие применимо практически ко всем API-функциям. Например, прототипы функций для библиотеки kernel32.dll находятся в заголовочном файле winbase.h и базовым типом для них является тип WINBASEAPI.

   Программа вызывается из командной строки следующим образом:

   `getproto <infile> <basetype>`, где

<infile> - заголовочный файл,
<basetype> - базовый тип.

Пример использования:

```
getproto winbase.h WINBASEAPI > winbase_protos.txt
```

В результате выполнения этой команды в файл winbase_protos.txt будут помещены прототипы функций из заголовочного файла winbase.h. . Выходные данные выводятся на консоль.

2) Программа CLEAR.

Эта программа очищает заголовочный файл от комментариев и отступов. Программа вызывается из командной строки следующим образом:

`clear <infile>`, где
<infile> - входной файл.

Пример использования:
clear winbase.h > winbase_clean.h

В результате выполнения этой команды в файл winbase_ clean.txt будет записан очищенный заголовочный файл winbase.h. . Выходные данные выводятся на консоль.

3) Программа GEN2_2.

Эта программа создаёт функции перехвата по имеющимся прототипам. Программа вызывается из командной строки следующим образом:

`gen2 <infile>`, где
<infile> - входной файл.
Выходные данные выводятся на консоль.

4) Программа DEFGET.

Эта программа извлекает из заголовочных файлов объявления типов (typedef) для дальнейшего разбора прототипов перехватываемых функций с помощью программы TD2DEF (см. п. 5)

Программа работает из командной строки. Синтаксис вызова:

`DEFGET <file.h>,` где file.h — входной файл. Выходные данные выводятся на консоль.

5) Программа TD2DEF.

   Эта программа на основе объявления типа (typedef) генерирует несколько макросов (define) (см. книгу Бьерна Страуструпа “Язык программирования C++”).

   Программа работает из командной строки. Синтаксис вызова:

   `TD2DEF <file.h>,`

   где file.h — входной файл. Выходные данные выводятся на консоль.

6) Файл DllGen.bat.

   Этот файл — генератор библиотек перехвата. Он на основе заголовочного файла создаёт библиотеку перехвата.

   Программа работает из командной строки. Синтаксис вызова:

   `DLLGEN <header.h> <basetype>,` где

   file.h — входной файл

   basetype — базовый тип (см. пп. 1).

   В результате работы командного файла создаётся cpp файл, который можно компилировать в среде Visual C++.

5.3. Установка программы

Распакуйте архив с программой в рабочую папку программы.

6. Выводы

1. Был разработан программный комплекс для создания библиотек перехвата.
2. Была создана библиотека перехвата для модуля USER32.DLL.

К сожалению, данная версия программного комплекса не полностью автоматизирует процесс создания библиотек перехвата и не исключает ручного редактирования, но разработки ведутся в направлении полной автоматизации этого процесса.

Применённая технология перехвата API функций позволяет создавать не только API-шпионов, но и может быть применена при создании различных эмуляторов, в частности эмуляторов реестра, файловой системы, аппаратных устройств и т. д. Это также является одним из перспективных направлений разработки.

7. Литература

1. Виктор Юров “Ассемблер. Специальный справочник” – Санкт-Петербург, изд. “Питер”, 2001
2. Бьерн Страуструп “Язык программирования C++” – Санкт-Петербург, изд. “Невский диалект”, 2002 г.
3. Чарльз Калверт “Программирование в Windows 95” – Москва, изд. “Бином”, 1996 г.

ПРИЛОЖЕНИЕ. ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММ

Файл Clear\CLEAR.CPP

```
//=============================================================================
// CLEAR.CPP
//=============================================================================

#define FALSE  0
#define TRUE   1

////////////////////////////////////////////////////////////////////////////

int     ClearReturns            (FILE * InFile, FILE * OutFile);
int     ClearComments           (FILE * InFile, FILE * OutFile);

//-----------------------------------------------------------------------------

void    DelShortComment         (FILE * File);
void    DelLongComment          (FILE * File);
void    DelComment              (FILE * InFile, FILE * OutFile);

//=============================================================================

int main (int argc, char * args[])
{
        clrscr ();
        if (argc < 2)
        {
                printf ("\n\nUSAGE : clear InFile [OutFile]\n");
                return 1;
        }

        FILE * InFile = NULL;
        FILE * OutFile = NULL;

        if (argc < 3)
                OutFile = stdout;
        el
                OutFile = fopen (args [2], "w");

        InFile = fopen (args [1], "r");

        if (!InFile)
        {
                printf ("Error opening file %s!\n", args [1]);
                return 1;
        }

        if (!OutFile)
        {
                printf ("Error opening file %s!\n", args [2]);
                return 1;
        }

        FILE * TmpFile = fopen ("$temp1", "w+");
        if (!TmpFile)
        {
                printf ("Error creating temp file!\n");
                return 1;
        }

        fseek (InFile, 0L, SEEK_SET);
        ClearComments   (InFile, TmpFile);
```

```
        fseek (TmpFile, 0L, SEEK_SET);
        ClearReturns   (TmpFile, OutFile);

    fcloseall ();
}

//=============================================================================

int ClearReturns (FILE * InFile, FILE * OutFile)
{
        if (!InFile || !OutFile)
                return FALSE;


        FILE * TmpFile = fopen ("$temp2", "w+");
        while (!feof (InFile))
        {
                char ch = getc (InFile);
                if (ch == '\r' ||
                        ch == '\t')
                {
                        ch = ' ';
                }
                putc (ch, TmpFile);
        }

        fseek (TmpFile, 0L, SEEK_SET);

        char ch1 = getc (TmpFile);
        char ch2 = getc (TmpFile);

        while (!feof (TmpFile))
        {
                while ( (ch1 == ' ' && ch2 == ' ') ||
                           (ch1 == '\n' && ch2 == '\n') )
                {
                        ch2 = getc (TmpFile);
         }

                putc (ch1, OutFile);

                ch1 = ch2;
                ch2 = getc (TmpFile);
        }
}

//=============================================================================

int ClearComments (FILE * InFile, FILE * OutFile)
{
        char ch1 = getc (InFile);
        while (!feof (InFile) )
        {
                if (ch1 == '/')
                {
                        DelComment (InFile, OutFile);
                }
                el
                {
                        putc (ch1, OutFile);
                }
                ch1 = getc (InFile);
        }
        return 1;
}

//=============================================================================

void DelComment (FILE * InFile, FILE * OutFile)
{
        char ch2 = getc (InFile);
```

```
        if (ch2 == '*')
        {
                DelLongComment (InFile);
        }
        else if (ch2 == '/')
        {
                DelShortComment (InFile);
        }
        else
        {
                putc ('/', OutFile);
                putc (ch2, OutFile);
                return;
        }
}

//===========================================================================

void DelLongComment (FILE * file)
{
        char ch1 = 0,
                ch2 = 0;

        while ((ch1 != '*' || ch2 != '/') && !feof (file))
        {
                ch1 = ch2;
                ch2 = getc (file);
        }
}

//===========================================================================

void DelShortComment (FILE * file)
{
        char ch = 0;
        while (ch != '\n' && !feof (file))
        {
                ch = getc (file);
        }
}

//===========================================================================
```

Файл Comment\CDEL.CPP

```
//===========================================================================
// CDEL.CPP
//===========================================================================

/////////////////////////////////////////////////////////////////////////////

void DelShortComment (FILE * File);
void DelLongComment (FILE * File);
void DelComment (FILE * InFile, FILE * OutFile);

//===========================================================================

int main (int argc, char* args[])
{
      if (argc < 2)
      {
            printf ("USAGE : cdel.exe infile.cpp [outfile.cpp]\n");
            return 1;
      }
      FILE * InFile  = fopen (args [1], "r");
      FILE * OutFile = NULL;

      if (argc < 3)
            OutFile = stdout;
```

```
      el
            OutFile = fopen (args [2], "w");

      clrscr ();
      char ch1 = getc (InFile);
      while (!feof (InFile) )
      {
            if (ch1 == '/')
            {
                  DelComment (InFile, OutFile);
            }
            el
            {
                  putc (ch1, OutFile);
            }
            ch1 = getc (InFile);
      }
      return 0;
}

//===========================================================================

void DelComment (FILE * InFile, FILE * OutFile)
{
      char ch2 = getc (InFile);
      if (ch2 == '*')
      {
            DelLongComment (InFile);
      }
      else if (ch2 == '/')
      {
            DelShortComment (InFile);
      }
      else
      {
            putc ('/', OutFile);
            putc (ch2, OutFile);
            return;
      }
}

//===========================================================================

void DelLongComment (FILE * file)
{
      char ch1 = 0,
             ch2 = 0;

      while ((ch1 != '*' || ch2 != '/') && !feof (file))
      {
            ch1 = ch2;
            ch2 = getc (file);
      }
}

//===========================================================================

void DelShortComment (FILE * file)
{
      char ch = 0;
      while (ch != '\n')
      {
            ch = getc (file);
      }
}

//===========================================================================
```

Файл DefGet\DEFGET.CPP

```
//=============================================================================
// DEFGET.CPP
//=============================================================================

#define MAXSTR          10000
#define TRUE            1
#define FALSE           0

/////////////////////////////////////////////////////////////////////////////

char * GetTypeDef ( FILE * File )
{
        if ( !File )
                return NULL;

        int Skobka = 0;
        int i = 0;

        char ch = getc ( File );
        if ( ch == EOF )
                return NULL;

        char * RetVal = new char [MAXSTR];
        if ( !RetVal )
                return NULL;

        memset ( RetVal, 0, MAXSTR );

        while ( ( ( ch != ';' ) || ( Skobka ) ) && ( ch != EOF ) )
        {
                if ( ( ch != '\n' ) && ( ch != '\r' ) && ( ch != '\t' ) )
                        RetVal [i++] = ch;

                if ( ch == '{' )
                        Skobka++;
                if ( ch == '}' )
                        Skobka--;

                ch = getc ( File );
        }
        return RetVal;
}

//=============================================================================

void StrDel ( char * str, char ch )
{
        int len = strlen (str);
        for ( int i = 0; i < len; i++ )
        {
                while ( ( str [i] == ch ) && ( str [i+1] == ch ) )
                {
                        for ( int k = i; k < len; k++ )
                        {
                                str [k] = str [k+1];
                        }
                        str [k+1] = 0;
                }
        }
}

//=============================================================================

void DelSpaces ( char * str )
{
        if ( !str )
                return;

        StrDel ( str, ' ' );
}
```

```
//============================================================================

int main ( int argsc, char * args[] )
{
        clrscr (  );

        for ( int i = 1; i < argsc; i++ )
        {
                FILE * File = fopen ( args [i], "r" );
                if ( !File )
                {
                        printf ( "\nÎøèáêà îòêðûòèÿ ôàéëà %s!\n", args[i] );
                        break;
                }

                char str [MAXSTR] = "";
                char * TDBody = NULL;
                int key = 0;

                if ( feof ( File ) )
                {
                        printf ( "\nÔàéë %s ïóñò!\n", args[i] );
                        break;
                }

                while ( ( !feof ( File ) ) && ( key < 3 ) )
                {
                        if ( kbhit (  ) )
                                key++;

                        while ( kbhit (  ) )
                                getch (  );

                        fscanf ( File, "%s", str );
                        if ( !strcmp ( str, "typedef" ) )
                        {
                                TDBody = GetTypeDef ( File );
                                if ( !TDBody )
                                        break;

                                DelSpaces ( TDBody );
                                printf ( "typedef %s;\n", TDBody );
                                delete TDBody;
                        }
                }
                fclose ( File );
        }
}
```

Файл DefGet\MYSTRTOK.H

```
//============================================================================
// MyStrtok.h
//============================================================================

#ifndef _MYSTRTOK
#define _MYSTRTOK

////////////////////////////////////////////////////////////////////////////

int     isin            (char ch, char* set);
char    __strtok        (char * set, char * dest);

char ** g_buffer = 0;

//----------------------------------------------------------------------------

char mystrtok (char** str, char* set, char* dest)
{
```

```
        if (str)
        {
                if (!dest)
                        return 0;

                g_buffer = str;
        }

        if (!g_buffer || !dest)
                return 0;

        return __strtok (set, dest);
}

//=========================================================================

char __strtok (char * set, char* dest)
{
        if (!**g_buffer)
        {
                return 0;
        }

        for ( ; (**g_buffer == ' ') && (**g_buffer); (*g_buffer)++ );
        for ( ; isin (**g_buffer, set) && (**g_buffer); (*g_buffer)++);

        int k = 0;
        while (!isin (**g_buffer, set) && **g_buffer)
        {
                dest [k++] = **g_buffer;
                (*g_buffer)++;

        }
        dest [k] = 0;
        char ch = **g_buffer;

        return ch;
}

//=========================================================================

int isin (char ch, char * set)
{
        for ( ; *set && set; set++)
        {
                if (ch == *set)
                        return 1;
        }
        return 0;
}
//=========================================================================

#endif //_STRTOK
```

Файл FilCat\FILCAT.CPP

```
//=========================================================================
// filcat.
//-------------------------------------------------------------------------
// (c) TiM!
//=========================================================================

/////////////////////////////////////////////////////////////////////////

int main (int argc, char * args [])
{
        if (argc < 2)
        {
                printf ("USAGE : %s infile\n", args [0]);
```

```
                return 1;
        }

        FILE * inFile = fopen (args [1], "r");

        if (!inFile)
        {
                printf ("Can't open file %s!\n", args [1]);
                return 1;
        }

        long int strn = 0;
        scanf ("%d", &strn);

        long int count = 0;
        while  (!feof (inFile))
        {
                char ch = getc (inFile);
                if (ch == '\n')
                        count++;
        }

        fseek (inFile, 0L, SEEK_SET);
        while (!feof (inFile))
        {
                char ch = getc (inFile);
                if (ch == '\n')
                        count--;
                if (count == strn)
                {
                        while (!feof (inFile))
                        {
                                ch = getc (inFile);
                                putc (ch, stdout);
                        }
                }
        }
}
```

Файл FillEn\FILLEN.CPP

```
//=============================================================================
// FILLEN.CPP
//-----------------------------------------------------------------------------
// (c) TiM!
//=============================================================================

int main (int argc, char * args [])
{
        if (argc < 2)
        {
                printf ("USAGE : %s \n", args [0]);
                return 1;
        }

        FILE * inFile = fopen (args [1], "r");
        if (!inFile)
        {
                printf ("Can't open file %s!\n", args [1]);
                return 1;
        }

        int count = 0;
        while (!feof (inFile))
        {
                char ch = getc (inFile);
                if (ch == '\n')
                        count++;
        }
```

```
        printf ("%d\n", count+1);
}
```

Файл FnGen\CARRAY.H

```
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// CArray class example
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

#ifndef _CARRAY
#define _CARRAY

#define STDSIZE 100

#define TRUE    1
#define FALSE   0

//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

template
class CArray
{
        public:

                // Constructor
                CArray                  ( void );
                CArray                  ( int size );
                CArray                  ( CArray & From );

                // Destructor
                ~CArray                 ( void );

                // Avaible operator
                CArray &                operator=       ( CArray & From );
                CArray &                operator=       ( char * str    );
                T &                     operator[]      ( int index     );
                int                     operator==      ( CArray & arr1 );
                int                     operator!=      ( CArray & arr1 );

                int                     strlen          ( void );
                int                     size            ( void );
                char *                  c_str           ( void );
                int                     strcpy          ( char * str );
                int                     strcat          ( char * str );
                int                     strcmp          ( char * str );

                int                     c_size;

        private:

                int                     cmparray ( CArray& arr1, CArray& arr2 );

                T *                     c_Container;

};

//============================================================================
// CArray implementation
//============================================================================

template
CArray::CArray ( void )
{
        c_Container = new T [ STDSIZE ];
        assert ( c_Container );
        c_size = STDSIZE;
        memset ( c_Container, 0, c_size );
}
```

```
//---------------------------------------------------------------------------

template
CArray::CArray ( int size )
{
        c_Container = new T [ size ];
        assert ( c_Container );
        c_size = size;
        memset ( c_Container, 0, c_size );
}

//---------------------------------------------------------------------------

template
CArray::CArray ( CArray & From )
{
        c_Container = new T [ From.c_size ];
        assert ( c_Container );
        memset ( c_Container, 0, c_size );

        c_size = From.c_size;

        for ( int i = 0; i < From.c_size; i++ )
        {
                c_Container [i] = From.c_Container [i];
        }
}

//---------------------------------------------------------------------------

template
CArray::~CArray ( void )
{
        if (c_Container)
                delete c_Container;
}

//---------------------------------------------------------------------------

template
int CArray::cmparray ( CArray & arr1, CArray & arr2 )
{
        int size1 = arr1.c_size;
        int size2 = arr2.c_size;

        if ( size1 != size2 )
                return 0;

        for ( int i = 0; i < size1; i++ )
        {
                if ( arr1.c_Container [i] != arr2.c_Container [i] )
                        return 0;
        }

        return 1;
}

//---------------------------------------------------------------------------

template
int CArray::size ( void )
{
        return c_size;
}

//---------------------------------------------------------------------------

int CArray::strlen ( void )
{
        return c_size;
}
```

```
//---------------------------------------------------------------------------

CArray & CArray::operator= ( char * str )
{
        c_size = ::strlen ( str ) + 1;

        if ( c_Container )
                delete c_Container;

        c_Container = new char [ c_size ];

        assert ( c_Container );
        ::strcpy ( c_Container, str );

        return *this;
}

//---------------------------------------------------------------------------

int CArray::strcmp ( char * str )
{
        return ::strcmp ( str, c_Container );
}

//---------------------------------------------------------------------------

int CArray::strcpy ( char * str )
{
        if ( !str )
                return FALSE;

        int len = ::strlen ( str );

        if ( ::strlen ( str ) > c_size )
        {
                delete c_Container;
                c_Container = new char [ len + 1 ];
        }

        if ( ::strcpy ( c_Container, str ) )
        {
                c_size = len + 1;
                return TRUE;
        }
        el
                return FALSE;

}

//---------------------------------------------------------------------------

int CArray::strcat ( char * str )
{
        if ( !str )
                return FALSE;

        int len1 = ::strlen ( str );
        int len2 = ::strlen ( c_Container );


        char * tmp = new char [len1 + len2 + 1];

        if ( !::strcpy ( tmp, c_Container ) )
        {
                delete tmp;
                return FALSE;
        }
        delete c_Container;

        c_Container = tmp;
        if ( !::strcat ( c_Container, str ) )
                return FALSE;
```

```
        return TRUE;
}

//---------------------------------------------------------------------------

template
CArray & CArray::operator= ( CArray & From )
{
        c_size = From.c_size;

        if ( c_Container )
                delete c_Container;

        c_Container = new T [ c_size ];

        assert ( c_Container );
        for ( int i = 0; i < c_size; i++ )
        {
                c_Container [i] = From.c_Container [i];
        }

        return *this;
}

//---------------------------------------------------------------------------

template
T & CArray::operator[] ( int index )
{
        assert ( c_Container );
        assert ( c_size >= index );

        return c_Container [index];
}

//---------------------------------------------------------------------------

template
int     CArray::operator== ( CArray & arr1 )
{
        return ( cmparray ( arr1, *this ) );
}

//---------------------------------------------------------------------------

template
int CArray::operator!= ( CArray & arr1 )
{
        return ( ! ( cmparray ( arr1, *this ) ) );
}

//---------------------------------------------------------------------------

template
char * CArray::c_str ( void )
{
        return c_Container;
}

//---------------------------------------------------------------------------

template
void ShowArray ( CArray ca1 )
{
        int size = ca1.SizeOf (  );

        for ( int i = 0; i < size; i++ )
        {
                printf ( "%d ", ca1 [i] );
        }
        printf ( "\n" );
```

```
}

//============================================================================

#endif
```

Файл FnGen\FNGEN.CPP

```
//============================================================================
//      GEN2.CPP
//----------------------------------------------------------------------------
// 07.01.03                     (c) TiM!
//============================================================================

#include "carray.h"
#include "template.h"

////////////////////////////////////////////////////////////////////////////

int G_pcount = 0;
#include "getfn.h"

//============================================================================

void Show (char ch)
{
        char c = ch;
        asm {
                mov ax, 0xb800
                mov es, ax
                xor bx, bx
//              inc bx
                mov al,
                mov byte ptr es:[bx], al
        }
}

//============================================================================

int main ( int Argc, char * Args[] )
{
        clrscr ();
        FILE * file = NULL;

        if ( Args [1] )
                file = fopen ( Args [1], "r" );
        el
        {
                printf ("\n\n"
                                "\nUsage:\n"
                                "GEN2_2 \n" );
                return 1;
        }

        if ( !file )
        {
                printf ( "Error opening file!\n" );
                return 1;
        }

        fseek ( file, 0, SEEK_SET );
        FN * fn = NULL;
        int k = 0,
                d = 0;

        char name       [MAXSTR]        = "",
                 type           [MAXSTR]        = "",
                 tmplt          [MAXSTR]        = "",
                 names          [BIGMAXSTR] = "",
```

```
                tmplts      [BIGMAXSTR] = "",
                asmcode     [BIGMAXSTR] = "",
                chars       [] = "|/-\\";

        while ( fn = GetFn (file) )
        {
                Show (chars [d++/1]);
                if (d > 4)
                        d = 0;

                int varsc = fn -> varsc;
                strcpy (tmplts, "");
                strcpy (names, "");
                strcpy (asmcode, "");
//--------------------------------------------------------------------------
                printf ( "extern \"C\" __declspec(dllexport) %s __stdcall %s(",
                                fn -> type -> c_str(),
                                fn -> name -> c_str() );

                for ( int i = 0; i < varsc-1; i++ )
                {
                        printf ( "%s %s,",
                                fn -> vars [i] -> type -> c_str(),
                                fn -> vars [i] -> name -> c_str() );
                }
                if ( varsc )
                        printf ( "%s %s",
                                fn -> vars [varsc-1] -> type -> c_str(),
                                fn -> vars [varsc-1] -> name -> c_str() );

                printf ( ")\n" );
                printf ( "{\n" );
                printf ( "      typedef %s __stdcall ptr (",
                                fn -> type -> c_str(),
                                fn -> vars [0] -> type -> c_str() );

                for ( i = 0; i < varsc-1; i++ )
                {
                        printf ( "%s,", fn -> vars [i] -> type -> c_str() );
                }

                if ( varsc )
                        printf ( "%s", fn -> vars [varsc-1] -> type->c_str());

                printf ( ");\n" );
                for ( i = 0; i < varsc; i++ )
                {
//--------------------------------------------------------------------------
                        strcpy ( name, fn -> vars [i] -> name -> c_str() );
                        strcpy ( type, fn -> vars [i] -> type -> c_str() );
                        GetParam ( name, type, tmplt );
/*                      if ( !GetParam ( name, type, tmplt ) )
                        {
                                printf ( "", name, type );
                                continue;
                        }
*/                      strcat ( names, ", " );
                        if ( !strcmp ( tmplt, "%s" ) )
                        {
                                strcat ( names, name );
                                strcat ( names, ", " );
                                strcat ( names, "(IsBadCodePtr (" );
                                strcat ( names, name );
                                strcat ( names, ") ? \"*** BAD PTR ***\" : " );
                                strcat ( names, name );
                                strcat ( names, " )" );
                                strcat ( tmplts, name );
                                strcat ( tmplts, " : (%p) \\\"%s\\\"; " );
                        }
                        el
                        {
                                if ( name [0] == '*' )
```

```
                                {
                                        strcat ( names, name+1 );
                                        strcat ( names, ", " );
                                        strcat ( names, "(IsBadCodePtr (" );
                                        strcat ( names, name+1 );
                                        strcat ( names, ") ? NULL : " );
                                        strcat ( names, name );
                                        strcat ( names, " )" );

                                        strcat ( tmplts, name+1 );
                                        strcat ( tmplts, " : (%p)" );
                                        strcat ( tmplts, tmplt );
                                        strcat ( tmplts, "; " );
                                }
                                else
                                {
                                        strcat ( names, name );
                                        strcat ( tmplts, name );
                                        strcat ( tmplts, " : " );
                                        strcat ( tmplts, tmplt );
                                        strcat ( tmplts, "; " );
                                }
                        }
                }
//------------------------------------------------------------------------
                printf ( "      static int FindStrCalled = 0,
                         "                 StrFound      = 0;
                         "      if ( !FindStrCalled )
                         "      {
                         "              StrFound = StrFind ( \"%s\" );
                         "              FindStrCalled = 1;
                         "      }
                         "      if ( StrFound )
                         "      {
                         "              dump (\"USER32 : %s \");
                         "              dump (\"(%s);\\n\"%s);
                         "      }
                         "      int  __hInst = LoadLibrary ( \"USER32.DLL\" );       \
                         "      int    proc  = GetProcAddress (__hInst, \"%s\");\n"
                         "      ptr*   stdfn = (ptr*)proc;
                        fn -> name -> c_str(),
                        fn -> name -> c_str(),
                        tmplts,
                        names,
                        fn -> name -> c_str(),
                        fn -> type -> c_str() );

                if ( strcmp (fn -> type -> c_str(), "void") )
                        printf ("       %s rez = stdfn (", fn-> type-> c_str());
                el
                        printf ("       stdfn (" );

                for ( i = 0; i < varsc-1; i++ )
                        printf ( "%s,", fn -> vars [i] -> name -> c_str() );
                if ( varsc )
                        printf ( "%s", fn -> vars [varsc-1]-> name-> c_str() );

                printf ( ");\n" );
                if ( strcmp (fn -> type -> c_str(), "void") )
                        printf ("       return rez;\n");


                printf ("}\n");
                printf
("//========================================================================\n" );
                delete fn;
                G_pcount = 0;
        }
}
```

Файл FnGen\GETFN.H

```
//=============================================================================
// GETFN.H
//-----------------------------------------------------------------------------
// 07.01.03                    (c) TiM!
//=============================================================================
#ifndef _GETFN
#define _GETFN

#include "carray.h"
#include "template.h"

////////////////////////////////////////////////////////////////////////////

#define         MAXSTR          100
#define         BIGMAXSTR       300
#define     TRUE                1
#define         FALSE           0
#define         MAXVARS         15

#define         DELETE(x)   if (x)                                      \
                                                {                       \
                                                        delete x;       \
                                                        x = NULL;       \
                                                }

typedef CArray String;

//=============================================================================

struct VAR
{
        String * name;
        String * type;

//-----------------------------------------------------------------------------

        VAR ()
        {
                name = NULL;
                type = NULL;
        }

//-----------------------------------------------------------------------------

        ~VAR ()
        {
                if ( name )
                        DELETE (name);
                if ( type )
                        DELETE (type);
        }

//-----------------------------------------------------------------------------

        dump ()
        {
                printf ( "VAR ( name = \"%s\"; type = \"%s\" );\n",
                                (name) ? name -> c_str() : "",
                                (type) ? type -> c_str() : "" );
        }

//-----------------------------------------------------------------------------

};

//=============================================================================

struct FN
```

```
{
        String * type;
        String * name;

        VAR * vars [MAXVARS];
        int varsc;

//---------------------------------------------------------------------------

        FN()
        {
                type = NULL;
                name = NULL;
                for ( int i = 0; i < MAXVARS; i++ )
                        vars [i] = NULL;
        }

//---------------------------------------------------------------------------

        ~FN()
        {
                if ( type )
                        DELETE (type);
                if ( name )
                        DELETE (name);
                for ( int i = 0; i < varsc && i < MAXVARS; i++ )
                        DELETE(vars [i]);
        }

//---------------------------------------------------------------------------

};

String * fGetStr ( FILE * File );
VAR * GetVar ( FILE * File );

////////////////////////////////////////////////////////////////////////////

int isin ( char Ch, char * Set )
{
        if ( !Set || !Ch )
                return FALSE;

        for ( int i = 0; i < strlen ( Set ); i++ )
        {
                if ( Ch == Set [i] )
                        return TRUE;
        }
        return FALSE;
}

//===========================================================================

char GetC (FILE * file)
{
        char ch = fgetc (file);
        ungetc (ch, file);

        return ch;
}

//===========================================================================

FN * GetFn ( FILE * File )
{
        FN * fn = new FN;
        assert ( fn );
        VAR * tmp = GetVar (File);
        if ( !tmp )
                return NULL;

        fn -> type = tmp -> type;
```

```
        fn -> name = tmp -> name;

        if ( GetC (File) == ')' )
        {
                fn -> varsc = 0;
                getc (File);
                return fn;
        }

        tmp = GetVar (File);
        int k = 0;
        while ( GetC (File) != ')' )
        {
                fn -> vars [k++] = tmp;
                tmp = GetVar (File);
                if ( !tmp )
                {
                        delete fn;
                        return NULL;
                }
        }
        fn -> vars [k++] = tmp;
        fn -> varsc = k;
        //getc (File);
        return fn;
}

//=============================================================================

VAR * GetVar ( FILE * File )
{
        VAR * var = new VAR;
        assert ( var );
        char ch = 0;

        String * str = fGetStr (File);
        if ( !str )
        {
                delete var;
                return NULL;
        }

        var -> type = str;

        if (isin (GetC (File), ",;{}()") )
        {
                var -> name = new String;
                char tmp [10] = {0};

                sprintf (tmp, "param%d", G_pcount++);
                var -> name -> strcpy (tmp);

                return var;
        }

        while ( 1 )
        {
                if ( !(str = fGetStr (File)) )
                        return NULL;

                if ( !isin (GetC (File), ",;{}()") )
                {
                        if ( str -> strlen() )
                                var -> type -> strcat ( " " );

                        var -> type -> strcat ( str -> c_str() );
                        delete str;
                }
                el
                {
                        var -> name = str;
                        break;
```

```
                }
        }
        char tmp [MAXSTR] = "";
        sprintf (tmp, "%s %s",
                        var -> type -> c_str(),
                        var -> name -> c_str());

        if (IsBaseType (tmp))
        {
                var -> type -> strcat (" ");
                var -> type -> strcat (var -> name -> c_str());
                sprintf (var -> name -> c_str(), "param%d", G_pcount++);
        }
        var -> name -> c_size = 8;
        return var;
}

//=========================================================================

String * fGetStr ( FILE * File )
{
        String * str = new String;
        assert ( str );
        char ch = getc (File);

        while ( isin ( ch, ",;{}()\n\r " ) )
                ch = getc (File);

        if ( ch == EOF )
                return NULL;

        int  k = 0;
        char * tmp = new char [ 100 ];
        assert ( tmp );
        memset ( tmp, 0, 100 );
        while ( ch && !isin ( ch, ",;(){}\n\r " ) && k < 100 )
        {
                if ( ch != ' ' );
                tmp [k++] = ch;
                ch = getc (File);
        }
        ungetc (ch, File);
        str -> strcpy ( tmp );
        DELETE (tmp);
        return str;
}

//=========================================================================

#endif
```

Файл FnGen\HEAD.CPP

```
// apispy2_user32_1.cpp : Defines the entry point for the DLL application.
//

//#include "stdafx.h"

#pragma warning ( disable : 4731 )
#define IMPORT __declspec(dllimport)
#define EXPORT __declspec(dllexport)

////////////////////////////////////////////////////////////////////

extern "C"
{
IMPORT int __stdcall LoadLibraryA     ( char* DllName );
IMPORT int __stdcall GetProcAddress   ( int hInst, char* lpProcName );
IMPORT int __stdcall IsBadCodePtr     ( char* ptr );
```

```
IMPORT int __stdcall _wsprintf        ( char* lpOutput, char* lpfmt, ... );
}

#define LoadLibrary LoadLibraryA

#define ESP_ADD0x4C

//=========================================================================

#define LOWORD(l)           ((int)((unsigned long)(l) & 0xffff))
#define HIWORD(l)           ((int)((unsigned long)(l) >> 16))
#define LOBYTE(w)           ((char)((unsigned long)(w) & 0xff))
#define HIBYTE(w)           ((char)((unsigned long)(w) >> 8))

//=========================================================================

int __stdcall DllMain( void* hModule,
                       unsigned long  ul_reason_for_call,
                       void* lpReserved
                                       )
{
        FILE * file = fopen ( "spy.txt", "a" );
        switch (ul_reason_for_call )
        {
                case 1
                        fprintf ( file, "**** Library attached! ***\n");
                break;
                case 2
                        fprintf ( file, "**** Library detached! ****\n" );
                break;
        }

        fclose ( file );
        return 1;
}

//=============================================================================

int StrFind ( char* str )
{
        char FileWord [100] = "";
        FILE * file = fopen ( "fns.dat", "r" );
        if ( !file )
                return NULL;

        while ( !feof (file) )
        {
                fscanf (file, "%s", FileWord);
                if ( !strcmp ( FileWord, str ) )
                {
                        fclose (file);
                        return 1;
                }
                getc (file);
        }
        return 0;
}

//=============================================================================

int dump( char *fmt, ... )
{
        va_list argptr;
        int cnt;

        FILE * file = fopen ("spy.txt", "a");
        va_start (argptr, fmt);
        cnt = vfprintf (file, fmt, argptr);
        va_end (argptr);
        fclose (file);

   return cnt;
```

```
}

//============================================================================
```

Файл FnGen\TEMPLATE.H

```
//============================================================================
// TEMPLATE.H
//============================================================================

#ifndef _TEMPLATE
#define _TEMPLATE

#define         MAXSTR  5000

#define         TRUE    1
#define         FALSE   0

////////////////////////////////////////////////////////////////////////////

char * GetTemplate ( char * type )
{

//----------------------------------------------------------------------------

        char Tmpls [][50] =
        {       "bool",                         "%d",
                "char",                 "%c",
                "signed char",          "%+c",
                "unsigned char",        "%c",
                "short",                "%hd",
                "signed short",         "%hd",
                "unsigned short",       "%hu",
                "short int",            "%hd",
                "signed short int",     "%hd",
                "unsigned short int",   "%hu",
                "int",                  "%d",
                "signed int",           "%d",
                "unsigned int",         "%u",
                "long",                 "%ld",
                "signed long",          "%ld",
                "unsigned long",        "%lu",
                "long int",             "%ld",
                "signed long int",      "%ld",
                "unsigned long int",    "%lu",
                "float",                "%g",
                "double",               "%g",
                "long double",          "%Lg" };

//----------------------------------------------------------------------------

        for ( unsigned long i = 0; i < sizeof(Tmpls)/50; i+= 2 )
        {
                if ( !strcmp ( type, Tmpls [i] ) )
                        return Tmpls [i+1];
        }
        return NULL;

//----------------------------------------------------------------------------

}

//============================================================================

int IsBaseType (char* type)
{
        while (type [strlen (type)-1] == '*')
                type [strlen (type)-1] = 0;
```

```
        return (GetTemplate (type)) ? 1 : 0;
}

//=============================================================================

int GetParam ( char * name, char * type, char * Template )
{
        if ( !strcmp (type, "void*") )
        {
                strcpy (Template, "%p");
                return TRUE;
        }
        if ( !strcmp (type, "char*") )
        {
                strcpy (Template, "%s");
                return TRUE;
        }

        if ( !name || !type )
                return FALSE;

        char temp [MAXSTR] = "";
        if ( type [strlen(type)-1] == '*' )
        {
                type [strlen(type)-1] = 0;
                strcpy (temp, "*");
                strcat (temp, name);
                strcpy (name, temp);
        }
        char * tmp = GetTemplate (type);
        if ( tmp )
        {
                strcpy (Template, tmp);
                return TRUE;
        }
        el
                return FALSE;
}

//=============================================================================

/*void main ( void )
{
        char name [] = "var";
        char type [] = "void*";
        char tmplt [MAXSTR] = "";
        GetParam ( name, type, tmplt );
        printf ( "\n%s %s %s\n", name, type, tmplt );
        char buffer [MAXSTR] = "";

        gets (buffer);
        printf ("%d\n", IsBaseType (buffer));
} */

#endif
```

Файл GetProto\GETPROTO.CPP

```
//=============================================================================
// getproto.
//-----------------------------------------------------------------------------
// (c) TiM!
//=============================================================================

#define         MAXSTR          500

////////////////////////////////////////////////////////////////////////////
```

```
int main (int argc, char * args [])
{
        if (argc < 3)
        {
                printf ("USAGE : getproto  \n");
                return 1;
        }

        FILE * inFile = fopen (args [1], "r");
        if (!inFile)
        {
                printf ("Can't open file %s\n", args [1]);
                return 1;
        }

        char    tmp [MAXSTR]    = {0};
        int     DefFlag                 = 0;

        while (!feof (inFile))
        {
                DefFlag = 0;
                fscanf (inFile, "%s", tmp);
                if (!strcmp (tmp, "#define"))
                {
                        DefFlag = 1;
                        fscanf (inFile, "%s", tmp);
                }
                if (!strcmp (tmp, args [2]) && !DefFlag)
                {
                        while (tmp [strlen (tmp) - 1] != ';' && !feof (inFile))
                        {
                                fscanf (inFile, "%s\n", tmp);
                                printf ("%s ", tmp);
                        }
                        printf ("\n");
                }
        }
}
```

Файл Patcher\PATCHER.CPP

```
//============================================================================
// Patcher.
//============================================================================

#define TL(a) tolower (a)

//============================================================================

long            flen            (FILE * file);
char *          fngets          (FILE * file, long from, int len, char *
buffer);

//----------------------------------------------------------------------------
// MAIN
//----------------------------------------------------------------------------
int main (int argc, char * args [])
{
        clrscr ();
        if (argc < 3)
        {
                printf ("USAGE : \n"
                                "patch  \n");
                return 1;
        }
        FILE * inFile = fopen (args [1], "r");
        FILE * outFile = fopen (args [2], "w");

        if (!inFile)
```

```
        {
                printf ("Can't open file %s\n", args [1]);
                return 1;
        }

        if (!outFile)
        {
                printf ("Can't open file %s\n", args [2]);
                return 1;
        }

        long inFileLen = flen (inFile);
        char * buffer = new char [inFileLen+1];

        char chrs [] = "|/-\\";
        char * LibName = "USER32.DLL";
        char FirstCh = LibName [0];
        int i = 0;

        int m = 0;

        while (!feof (inFile))
        {
                char ch = getc (inFile);
                if (TL (ch) == TL (FirstCh))
                {
                        long curpos = ftell (inFile);

                        ungetc (FirstCh, inFile);
                        for (i = 0; i < strlen (LibName); i++)
                        {
                                ch = getc (inFile);
                                if (TL (ch) != TL (LibName [i]))
                                        break;
                        }
                        if (i == strlen (LibName))
                                printf ("FOUND : %s at 0x%X\n", LibName,curpos);

//                      fseek (inFile, curpos, SEEK_SET);
                }
        }


        delete buffer;
}

//---------------------------------------------------------------------------
// flen
//---------------------------------------------------------------------------

long flen (FILE * file)
{
        if (!file)
                return 0;

        long curpos  = ftell (file);
        fseek (file, 0L, SEEK_END);
        long len = ftell (file);

        fseek (file, curpos, SEEK_SET);
        return len;
}

//---------------------------------------------------------------------------
// fnget
//---------------------------------------------------------------------------

char * fngets (FILE * file, long from, int len, char * buffer)
{
        long curpos = ftell (file);
        fseek (file, from, SEEK_SET);
```

```
        if (!buffer)
                return NULL;

        int k = 0;
        for (int i = 0; i < len; i++)
        {
                buffer [k++] = getc (file);
        }
        buffer [k] = 0;

        fseek (file, curpos, SEEK_SET);
        return buffer;
}
```

Файл TD2Def\MYSTRTOK.H

```
//=============================================================================
// MyStrtok.h
//=============================================================================

#ifndef _MYSTRTOK
#define _MYSTRTOK

/////////////////////////////////////////////////////////////////////////////

int     isin            (char ch, char* set);
char    __strtok        (char * set, char * dest);

char ** g_buffer = 0;

//-----------------------------------------------------------------------------

char mystrtok (char** str, char* set, char* dest)
{
        if (str)
        {
                if (!dest)
                        return 0;

                g_buffer = str;
        }

        if (!g_buffer || !dest)
                return 0;

        return __strtok (set, dest);
}

//=============================================================================

char __strtok (char * set, char* dest)
{
        if (!**g_buffer)
        {
                return 0;
        }

        for ( ; (**g_buffer == ' ') && (**g_buffer); (*g_buffer)++ );
        for ( ; isin (**g_buffer, set) && (**g_buffer); (*g_buffer)++);

        int k = 0;
        while (!isin (**g_buffer, set) && **g_buffer)
        {
                dest [k++] = **g_buffer;
                (*g_buffer)++;

        }
        dest [k] = 0;
```

```
        char ch = **g_buffer;

        return ch;
}

//============================================================================

int isin (char ch, char * set)
{
        for ( ; *set && set; set++)
        {
                if (ch == *set)
                        return 1;
        }
        return 0;
}

//============================================================================

#endif //_STRTOK
```

Файл TD2Def\TD2DEF.CPP

```
//============================================================================
// TD2DEF.CPP
//----------------------------------------------------------------------------
// (c) TiM!
//============================================================================

#include "mystrtok.h"

#define   MAXSTR 2500

////////////////////////////////////////////////////////////////////////////

void            GetDefine               (char * strIn                           );
void            GetSType                (char * p, char * lastword    );
void            GetStruct               (char * p                               )
void            clear                   (char * str                             )
void            GetFnProto              (char * p                               )
void            trace                   (char * str                             )

//============================================================================
// main
//============================================================================

int main (int argc, char* args [])
{
        char TDstr [MAXSTR] = {0};
        clrscr ();

        FILE * file = fopen (args [1], "r");
        assert (file);

        int k = 0;
        char ch = ' ';

        while (!feof (file))
        {
                while (ch != '\n' && k < MAXSTR && !feof (file))
                {
                        ch = getc (file);
                        TDstr [k++] = ch;
                }
                TDstr [k]   = 0;

                trace ("\n//");
                trace (TDstr);
                trace ("\n");
```

```
                GetDefine (TDstr);
                k = 0;
                ch = ' ';
        }
        fclose (file);
}

//=============================================================================
// GetDefin
//=============================================================================

void GetDefine (char* strIn)
{
        clear (strIn);
        strIn [strlen (strIn)] = ';';

        char * p = strIn;
        char   buffer [MAXSTR] = {0};

        char    ch              = 1;
        int     k       = 0;

        ch = mystrtok (&p, ",;{}() ", buffer);

        if (strcmp (buffer, "typedef"))
        {
                return;
        }

        ch = mystrtok (NULL, ",;{}() ", buffer);

        if (!strcmp (buffer, "struct"))
        {
                GetStruct (p);
        }
//-----------------------------------------------------------------------------
        else if (!strcmp (buffer, "union"))
        {
                GetStruct (p);
        }
//-----------------------------------------------------------------------------
        else if (!strcmp (buffer, "enum"))
        {
                printf ("//\n");
        }
//-----------------------------------------------------------------------------
        else
        {
                while ((p [k] != ',') && (p [k] != ',') && (p [k]))
                {
                        if (p [k++] == '(')
                        {
                                GetFnProto (p);
                                return;
                        }
                }
                GetSType (p, buffer);
        }
}

//=============================================================================
// GetStruct
//=============================================================================

void GetStruct (char * p)
{
        int skobka              = 0;
        char ch                 = ' ';
        char buffer [MAXSTR]    = {0};
        int strtok_init         = 0;
```

```
        for ( ;*p == ' '; p++);
        if (*p != '{')
        {
                ch = mystrtok (&p, ",;{}() ", buffer);
                printf ("#define %s void\n", buffer);
                strtok_init = 1;
        }

        char * tmp = p;
        while (*(p++) != '{' && (*p));
        if (!*p)
        {
                p = tmp;
        }
        el
        {
                skobka++;

                while (skobka && *p)
                {
                        if (*p == '{') skobka++;
                        if (*p == '}') skobka--;

                        p++;
                }
        }

        if (!*p)
                return;

        clear (buffer);

        while ((ch != ';') && (*p) && (ch))
        {
                if (!strtok_init)
                        ch = mystrtok (&p, ",;{}() ", buffer);
                el
                        ch = mystrtok (NULL, ",;{}() ", buffer);

                clear (buffer);

                if (buffer [0] == '*')
                        printf ("#define %s void*\n", buffer+1);
                el
                        printf ("#define %s void\n", buffer);

        }
}

//=============================================================================
// GetSTy
//=============================================================================

void GetSType (char * p, char * lastword)
{
        char buffer [MAXSTR] = {0};
        char from   [MAXSTR] = {0};
        char to     [MAXSTR] = {0};

        char ch = ' ';

        strcpy (from, lastword);

        ch = mystrtok (&p, ",;{}() ", buffer);

        if ((ch != ',' && ch != ';') && (*p))
        {

                strcat (from, " ");
                strcat (from, buffer);

                while (ch != ',' && ch)
```

```
                {
                        ch = mystrtok (NULL, ",;{}() ", buffer);
                        if ((ch == ',' || ch == ';') && (*p))
                                break;

                        strcat (from, " ");
                        strcat (from, buffer);
                }
        }
        clear (buffer);

        if (buffer [0] == '*')
                printf ("#define %s %s*\n", buffer+1, from);
        el
                printf ("#define %s %s\n", buffer, from);


        while ((ch != ';') && (*p) && (ch))
        {
                ch = mystrtok (NULL, ",;{}() ", buffer);
                clear (buffer);

                if (buffer [0] == '*')
                        printf ("#define %s %s*\n", buffer+1, from);
                el
                        printf ("#define %s %s\n", buffer, from);
        }
}

//=============================================================================
// GetFnProto
//=============================================================================

void GetFnProto (char * p)
{
        printf ("//FnProto\n");
}

//=============================================================================
// clear
//=============================================================================

void clear (char* str)
{
        if ( (str [strlen (str)-1] == ',') ||
                 (str [strlen (str)-1] == ';')  )
        {
                str [strlen (str)-1] = 0;
        }

        for (int k = 0; k < strlen (str); k++)
        {
                while (str [k] == ' ' && str [k+1] == ' ')
                {
                        for (int i = k+1; i < strlen (str); i++)
                        {
                                str [i] = str [i+1];
                        }
                        str [i] = 0;
                }
        }
}

//=============================================================================
// tr
//=============================================================================

void trace (char * str)
{
        fprintf (stdout, "%s", str);
}
```

ПОПОВ АРТЁМ, ЛИЦЕЙ 1580, 11 КЛАСС

Слайд 1 **(цели и задачи)**

УВАЖАЕМАЯ КОМИССИЯ, **РАЗРЕШИТЕ ПРЕДСТАВИТЬ** Вам проект — Систему перехвата WIN32 API функций.

Этот **ПРОГРАММА** задумывалась как **УНИВЕРСАЛЬНАЯ** система, позволяющая **ПЕРЕХВАТИТЬ ЛЮБЫЕ ФУНКЦИИ**, вызываемые WIN32 приложениями из динамически подключаемых модулей (DLL).

Было **РАССМОТРЕНО МНОЖЕСТВО** способов перехвата, но **ВЫБРАН** был самый **ПОДХОДЯЩИЙ** для решения данной задачи.

Слайд 2 **(структ. win32 исп. файла)**

ПЕРЕХВАТ основывается на **МОДУЛЬНОСТИ АРХИТЕКТУРЫ** операционной системы WINDOWS. **БОЛЬШИНСТВО ПРОГРАММ**, работающих в этой операционной системе **ИСПОЛЬЗУЮТ** те или иные системные **МОДУЛИ**, которые **РЕАЛИЗОВАНЫ** в виде динамически подключаемых **DLL-БИБЛИОТЕК**. Структура WIN32 исполняемого файла позволяет подключать к программе новые модули или заменять существующие.

WIN32 исполняемый файл состоит из **МНОЖЕСТВА СЕКЦИЙ**.

Рассмотрим **ТАБЛИЦУ ИМПОРТА**, так как с помощью неё к программе **ПОДКЛЮЧАЮТСЯ ВНЕШНИЕ МОДУЛИ**, **ФУНКЦИИ** которых и **ПЕРЕХВАТЫВАЮТСЯ** программой.

СЕКЦИЯ ИМПОРТА содержит **ДАННЫЕ О ФУНКЦИЯХ**, вызываемых из внешних модулей. В ней хранятся **ИМЕНА МОДУЛЕЙ**, **ИНФОРМАЦИЯ** о самих функциях, и **АДРЕСА** этих функций. При работе программы, функции **ВЫЗЫВАЮТСЯ** из модуля, **ИМЯ** которого содержится в таблице импорта. Таким образом, **ИЗМЕНЯЯ ИМЯ** стандартного модуля на **ИМЯ МОДУЛЯ ПЕРЕХВАТА**, мы по сути **ПОДМЕНЯЕМ СТАНДАРТНЫЕ ФУНКЦИИ** на **ФУНКЦИИ ПЕРЕХВАТА**. Такой модуль перехвата реализован в dll-библиотеке, которая называется **БИБЛИОТЕКОЙ ПЕРЕХВАТА**. Рассмотрим схему её работы.

Слайд 3 **(схема работы библиотеки перехвата)**

Для того чтобы библиотека перехвата работала, её **НЕОБХОДИМО ПОДКЛЮЧИТЬ** к исполняемому файлу путём **ЗАМЕНЫ ИМЕНИ** стандартного модуля на имя модуля перехвата.

После запуска программы, она начинает **ВЫЗЫВАТЬ ФУНКЦИИ** из **ВНЕШНИХ МОДУЛЕЙ**. Если бы к программе **НЕ БЫЛ ПОДКЛЮЧЁН ПЕРЕХВАТЧИК**, то функции вызывались бы **НАПРЯМУЮ** из стандартного модуля. На слайде такой слу-

чай показан пунктирными стрелками. Когда **ПЕРЕХВАТЧИК ПОДКЛЮЧЁН**, то функции **ВЫЗЫВАЮТСЯ** из **БИБЛИОТЕКИ ПЕРЕХВАТА**. Такие функции называются **ФУНКЦИЯМИ ПЕРЕХВАТА**. Они могут выполнять **РАЗЛИЧНЫЕ ДЕЙСТВИЯ**. Например, просто **ВОЗВРАЩАТЬ УПРАВЛЕНИЕ** в программу, если с помощью перехватчика нужно **ЗАБЛОКИРОВАТЬ** вызов стандартной функции, или она может сохранять информацию о вызванной функции в файл. Именно так и работают функции перехвата в программе APISpy32. Рассмотрим подробнее **АЛГОРИТМ РАБОТЫ** такой функции.

Слайд 4 **(алгоритм работы функции перехвата)**

Сначала происходит **ПОИСК ИМЕНИ** функции в специальном файле на диске. Если имя **НАЙДЕНО**, то на диск **ЗАПИСЫВАЕТСЯ ИНФОРМАЦИЯ** о функции. Затем **ВЫЗЫВАЕТСЯ СТАНДАРТНАЯ ФУНКЦИЯ** с теми же параметрами, что и функция перехвата. После этого **УПРАВЛЕНИЕ ВОЗВРАЩАЕТСЯ** в вызывающую программу.

Таким образом, в **РЕЗУЛЬТАТЕ** работы модуля перехвата на диске **СОЗДАЁТСЯ ФАЙЛ С ОТЧЁТОМ О ВЫЗВАННЫХ ФУНКЦИЯХ**.

Слайд 5 **(пример отчёта билиотеки перехвата)**

Отчёт о вызванных функциях **СОСТОИТ ИЗ СТРОК**, **КАЖДАЯ** из которых **ОТВЕЧАЕТ ЗА ОДИН ВЫЗОВ** функции. Такой вызов **ОПИСЫВАЕТСЯ ВРЕМЕНЕМ,** прошедшим с момента запуска прогрммы, **ИМЕНЕМ** стандартного модуля, в котором находится функция, **ИМЕНЕМ ФУНКЦИИ**, и **ИНФОРМАЦИЕЙ О ПАРАМЕТРАХ**.

ИНФОРМАЦИЯ О ПАРАМЕТРЕ может быть **НЕСКОЛЬКИХ ТИПОВ**.

1. Если параметр – **УКАЗАТЕЛЬ НА СТРОКУ**, то сохраняется **ЗНАЧЕНИЕ САМОГО УКАЗАТЕЛЯ** (оно в скобках) и, по возможности, **САМА СТРОКА**.
2. Если параметр – **УКАЗАТЕЛЬ ДРУГОГО ВИДА**, то сохраняется **ЗНАЧЕНИЕ УКАЗАТЕЛЯ** и **ДАННЫЕ**, находящиеся по указанному адресу.
3. Для всех **ДРУГИХ ТИПОВ** параметров **СОХРАНЯЕТСЯ** только **ЗНАЧЕНИЕ** самого **ПАРАМЕТРА**.

Слайд 6 **(результаты работы)**

При выполнении данной работы достигнуты следующие результаты:

1) Разработан программный комплекс для создания библиотек перехвата
2) Созданы библиотеки перехвата для модулей USER32 и GDI32

Описаную технологию можно применить в различных программах.

Можно разработать множество эмуляторов, работа которых основана на перехвате системных функций Windows, например эмулятор реестра или эмулятор файловой системы.

Также будет производиться работа по доработке системы перехвата, а именно будет дорабатываться генератор библиотек перехвата и будут созданы библиотеки перехвата для основных системных модулей.

Теперь **ПЕРЕЙДЕМ К ДЕМОНСТРАЦИИ,** а по ходу я могу ответить на ваши вопросы.

ЗАПУСК

Запустим редактор реестра с подключёнными модулями перехвата для системных библиотек USER32.DLL и GDI32.DLL

<Запуск bin\examples\regedit\regedit>

Выполним различные действия.

<Немного поработать, закрыть regedit>

Как видим, создался файл spy.txt. Это – файл отчёта.

<Открыть spy.txt>

Отключим перехват какой нибудь функции, например SendMessage.

<Закрыть spy.txt, открыть fns.dat>

Для этого удалим её имя из файла fns.dat.

<Найти строки SendMessageA и SendMessageW и удалить их>

<Сохранить изменения>

Удалим старый отчёт.

<Удалить spy.txt>

Запустим программу снова.

<Запуск bin\examples\regedit\regedit>

Выполним различные действия.

<Немного поработать, закрыть regedit>

<Открыть spy.txt>

<Поиском найти строку SendMessage>

Как видим строка не найдена.

<Закрыть всё>

ЗАКЛЮЧЕНИЕ

В заключении хотелось бы **поблагодарить** своего **РУКОВОДИТЕЛЯ –** Дединского Илью Рудольфовича, а также доцента кафедры ИУ4 МГТУ им. Баумана Власова Андрея Игоревича за помощь в подготовке доклада.

СПАСИБО ЗА ВНИМАНИЕ.