Выполнила: Белоусова Е., ИП-911

Задача

Цель: приобрести навыки разработки приложений с использованием dll и освоить интерфейс PSAPI.
Задание 1: написать программу <prog1>, с динамической загрузкой <lib1>.dll.
Задание 2: написать программу <mon1> с использованием функций PSAPI EnumProcesses, OpenProcess, EnumProcessModules, GetModuleBaseName и GetModuleFileName, определяющую все модули процесса и места расположения, соответствующих им файлов.
Задание 3: запустить <prog1>, периодически загружая и выгружая библиотеку < lib1>.dll; используя программу < mon1> отследить наличие модуля <lib1> в процессе <prog1>.

Описание работы программы

Реализуем в библиотеке lib1 функции для возведения в квадрат, суммы и вычитания.

Скомпилируем lib1.dll с помощью команды *cl /LD lib1.c*.

Динамически загрузим lib1.dll с помощью функции LoadLibrary. Функция ищет образ dll-файла и пытается спроецировать его на адресное пространство вызывающего процесса. Возвращаемое значение сообщает адрес виртуальной памяти, по которому спроецирован образ файла, либо null, если спроецировать не удалось.

```
hInst = LoadLibrary("lib1.dll");
if((hInst = LoadLibrary("lib1")) == NULL)
{
    return 1;
}
pfnMyFunction = (fSquare)GetProcAddress(hInst, "square");
iCode = (*pfnMyFunction)(3);
printf("%d\n", iCode);

pfnMy = (func)GetProcAddress(hInst, "add");
iCode = (*pfnMy)(3,5);
printf("%d\n", iCode);

pfnMy = (func)GetProcAddress(hInst, MAKEINTRESOURCE(3));
iCode = (*pfnMy)(5,3);
printf("%d\n", iCode);
```

Попробуем по-разному обратиться к функциям: по имени и по порядковому номеру, который был указан в .def через @.

```
PS D:\семестр 5\os\лаб7> ./prog1.exe
9
8
2
PS D:\семестр 5\os\лаб7>
```

В программе mon1 с помощью функции EnumProcesses получим список идентификаторов процессов. Распечатаем все модули процесса с помощью функции PrintProcessModules, передав идентификатор нужного процесса.

Распечатаем модули последнего процесса (запуск самой программы mon1):

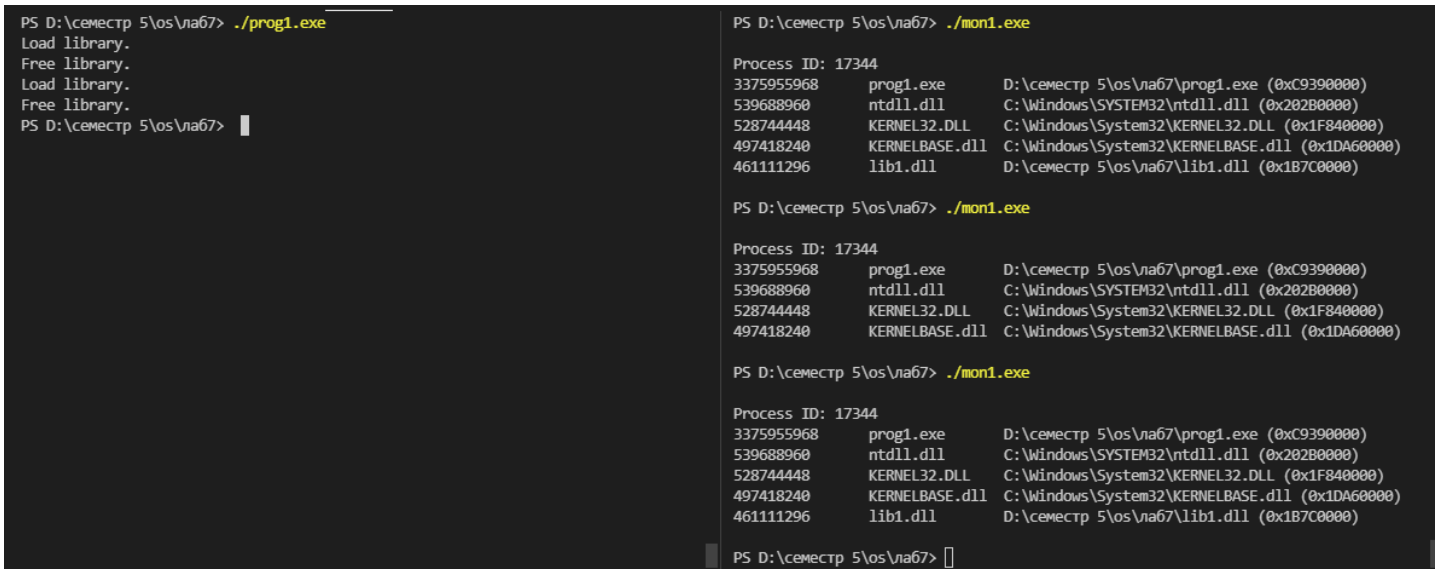
```
    PrintProcessModules(aProcesses[cProcesses - 1]);
    printf("\n");
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Copyright (C) Microsoft Corporation.  All rights reserved.

/out:mon1.exe
mon1.obj
PS D:\семестр 5\os\лаб7> ./mon1.exe

Process ID: 16012
4166713344      mon1.exe        D:\семестр 5\os\лаб7\mon1.exe (0xF85B0000)
539688960       ntdll.dll       C:\Windows\SYSTEM32\ntdll.dll (0x202B0000)
528744448       KERNEL32.DLL    C:\Windows\System32\KERNEL32.DLL (0x1F840000)
497418240       KERNELBASE.dll  C:\Windows\System32\KERNELBASE.dll (0x1DA60000)
```

Распечатаем модули Internet Explorer (передадим идентификатор предпоследнего процесса):

```
PS D:\семестр 5\os\лаб7> ./mon1.exe

Process ID: 3980
9109504 IEXPLORE.EXE    C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE (0x008B0000)
539688960       ntdll.dll       C:\Windows\SYSTEM32\ntdll.dll (0x202B0000)
529530880       wow64.dll       C:\Windows\System32\wow64.dll (0x1F900000)
538836992       wow64win.dll    C:\Windows\System32\wow64win.dll (0x201E0000)
1997864960      wow64cpu.dll    C:\Windows\System32\wow64cpu.dll (0x77150000)
```

Запустим <prog1>, периодически загружая и выгружая библиотеку <lib1>.dll; используя программу < mon1> отследим наличие модуля <lib1> в процессе <prog1>.



Листинг

```
//lib1.c
#include <windows.h>
#define DllExport __declspec(dllexport)
#include "lib1.h"
int square(int a)
{
   return a * a;
}

int add(int a, int b)
{
   return a+b;
}

int sub(int a, int b)
{
   return a-b;
}

//lib1.h
#pragma once
#define DllExport __declspec(dllexport)
DllExport int square(int a);
DllExport int add(int a, int b);
DllExport int sub(int a, int b);

//lib1.def
LIBRARY lib1
EXPORTS
   square @1
   add @2
   sub @3
//prog1.c
#include <windows.h>
#include <conio.h>
```

```
typedef int (*fSquare)(int);
typedef int (*func)(int, int);

int main()
{
   HINSTANCE hInst;
   func pfnMy;
   fSquare pfnMyFunction;
   int iCode;
   while(!kbhit())
   {
      // char esc = getch();
      // if(esc == 27)
      // {
      //    break;
      // }
      if((hInst = LoadLibrary("lib1.dll")) == NULL)
      {
         return 1;
      }
      printf("Load library.\n");
      Sleep(5000);
      // getch();
      // if(esc == 27)
      // {
      //    break;
      // }
      boolean fFreeResult;
      if (hInst != NULL)
      {
         fFreeResult = FreeLibrary(hInst);
      }
      printf("Free library.\n");
      Sleep(6000);
   }
   // hInst = LoadLibrary("lib1.dll");
   // if((hInst = LoadLibrary("lib1")) == NULL)
   // {
   //    return 1;
   // }
   // pfnMyFunction = (fSquare)GetProcAddress(hInst, "square");
   // iCode = (*pfnMyFunction)(3);
   // printf("%d\n", iCode);

   // pfnMy = (func)GetProcAddress(hInst, "add");
   // iCode = (*pfnMy)(3,5);
   // printf("%d\n", iCode);

   // pfnMy = (func)GetProcAddress(hInst, MAKEINTRESOURCE(3));
   // iCode = (*pfnMy)(5,3);
   // printf("%d\n", iCode);
   return 0;
}
```

```
//mon1.c
#include <windows.h>
#include "Psapi.h"

void PrintProcessModules(DWORD pID)
{
   HMODULE modHndls[1024];
   HANDLE pHndle;
   DWORD cbNeeded;
   unsigned int i;
   printf("\nProcess ID: %u\n", pID);
   pHndle = OpenProcess(PROCESS_QUERY_INFORMATION | PROCESS_VM_READ,
                        FALSE, pID);
   if (NULL == pHndle)
      return;
   if(EnumProcessModules(pHndle, modHndls, sizeof(modHndls), &cbNeeded))
   {
      for (i = 0; i < (cbNeeded / sizeof(HMODULE)); i++)
      {
         char szModName[MAX_PATH];
         GetModuleBaseName(pHndle, modHndls[i], (LPSTR)szModName, sizeof(szModName));
         printf("%u\t%s", modHndls[i],szModName);
         if (GetModuleFileNameEx(pHndle, modHndls[i], szModName, sizeof(szModName)))
         {
            printf("\t%s (0x%08X)\n", szModName, modHndls[i]);
         }
      }
   }
   CloseHandle( pHndle );
}

void main( )
{
   SetConsoleOutputCP(1251);
   DWORD aProcesses[1024], cbNeeded, cProcesses;
   unsigned int i;
   if (!EnumProcesses(aProcesses, sizeof(aProcesses), &cbNeeded))
      return;

   cProcesses = cbNeeded / sizeof(DWORD);
   PrintProcessModules(aProcesses[cProcesses - 2]);
   printf("\n");
}
```