

## Работа 1. Исследование объектов Windows

**Цель работы:** исследовать объектные механизмы Win32.

**Задание 1.1.** Получить список отрытых объектов и изучить типы объектов.

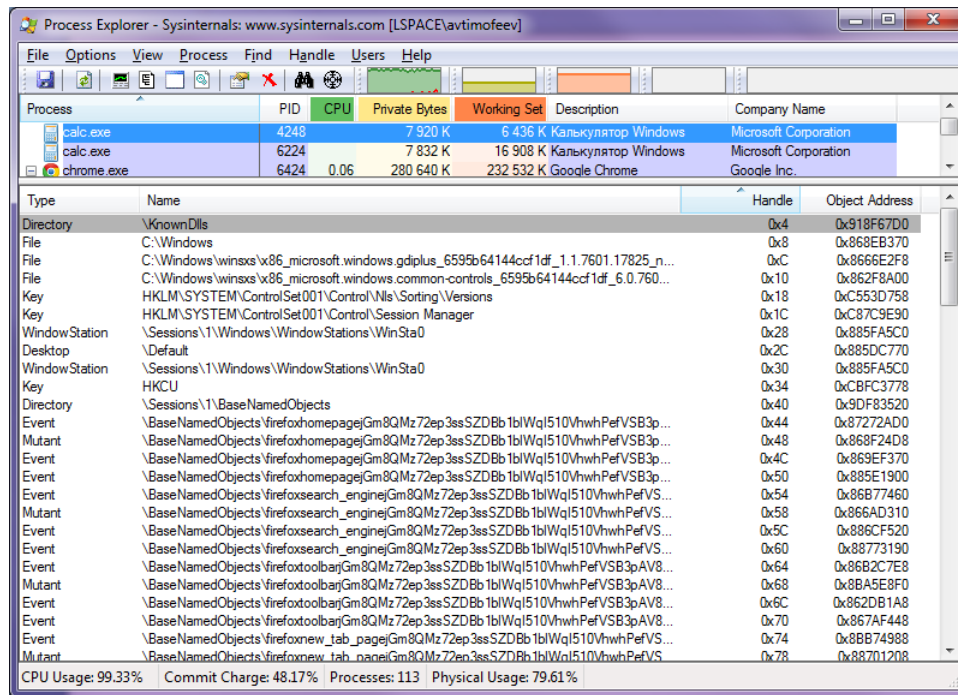
Утилита *Process Explorer* заменяет *Диспетчер задач Windows*, отображая более подробную информацию о процессах и потоках, включая их родство, загруженные DLL и открытые дескрипторы объектов.

*Handle* – консольная утилита, отображающая информацию о дескрипторах объектов, имеющихся у процессов системы. Если запустить утилиту без параметров командной строки, она выведет список всех процессов и все описатели файлов и именованных разделов, занятых этими процессами, разделяя данные о каждом процессе пунктирной линией. Для каждого процесса отображается имя, PID и имя учетной записи, от имени которой запущен процесс. Далее указываются описатели, принадлежащие этому процессу.

Для получения справочной информации по синтаксису командной строки утилиты можно набрать команду *handle -h*.

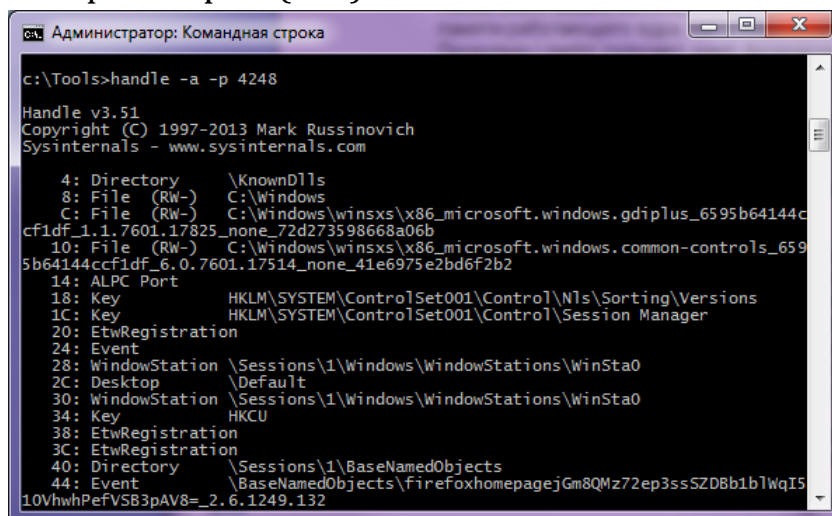
### Указания к выполнению.

1. Скачайте и распакуйте в каталог **c:\Tools** утилиту *Process Explorer* (<http://technet.microsoft.com/ru-ru/sysinternals/bb896653.aspx>).
2. Выполните запуск утилиты *Process Explorer* от имени администратора.
3. В интерфейсе *Process Explorer* выберите процесс для исследования и ознакомьтесь со списком объектов, принадлежащих выбранному процессу. Запишите в отчет результаты выполнения данного шага. На приведенном ниже рисунке в качестве примера выбран процесс *calc.exe* с идентификатором (PID) 4248<sub>10</sub>. Обратите внимание, что значение дескриптора объекта всегда кратно 4, почему это так мы рассмотрим в следующем задании.



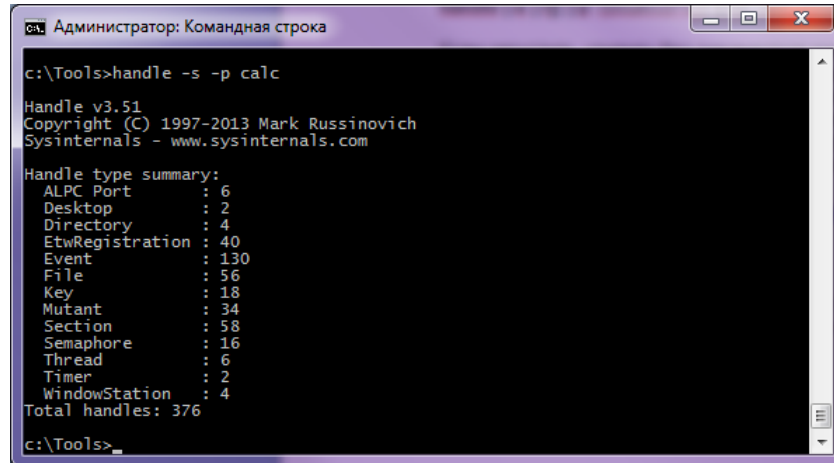
4. Скачайте и распакуйте в каталог **c:\Tools** утилиту *Handle* (<http://technet.microsoft.com/ru-ru/sysinternals/bb896655>).

5. Запустите командную строку от имени администратора, перейдите в каталог **c:\Tools** и запустите утилиту *Handle.exe* со следующими параметрами: *handle -a -p <PID>*, где *PID* – идентификатор процесса, выбранного в пункте 3 этого задания. В результате Вы получите список объектов, принадлежащих выбранному процессу. Запишите в отчет результаты выполнения данного шага, сравните полученные сведения с результатами выполнения пункта 3. На приведенном ниже рисунке представлен перечень объектов процесса *calc.exe* с идентификатором (PID) 4248<sub>10</sub>.



6. Запустите утилиту *Handle.exe* со следующими параметрами: *handle -s -p <PID>*, где *PID* – идентификатор, выбранного в пункте 3

процесса. В результате Вы получите сводный список с количеством объектов разного типа, принадлежащих выбранному процессу. Запишите в отчет результаты выполнения данного шага, с помощью Интернет выясните назначение этих типов объектов и прокомментируйте их в отчете.



```

c:\Tools>handle -s -p calc

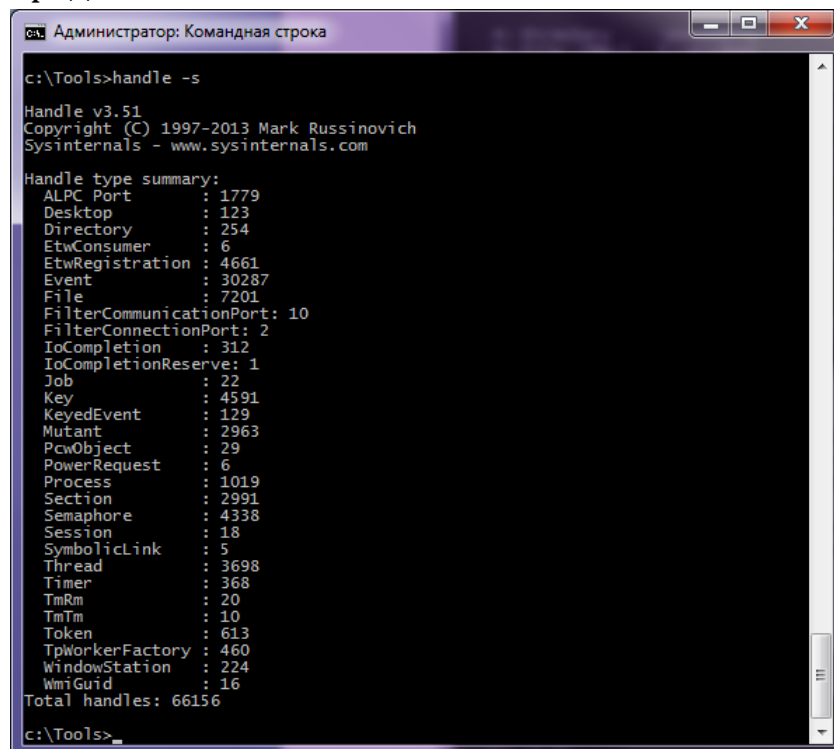
Handle v3.51
Copyright (C) 1997-2013 Mark Russinovich
Sysinternals - www.sysinternals.com

Handle type summary:
ALPC Port      : 6
Desktop       : 2
Directory     : 4
EtwRegistration : 40
Event         : 130
File          : 56
Key           : 18
Mutant        : 34
Section       : 58
Semaphore     : 16
Thread        : 6
Timer         : 2
WindowStation : 4
Total handles: 376

c:\Tools>

```

7. Запустите утилиту *Handle.exe* со следующими параметрами: *handle -s*. В результате Вы получите сводный список с количеством объектов разного типа для всех процессов вычислительной системы. Запишите в отчет результаты выполнения данного шага, поясните назначение представленных в отчете типов объектов.



```

c:\Tools>handle -s

Handle v3.51
Copyright (C) 1997-2013 Mark Russinovich
Sysinternals - www.sysinternals.com

Handle type summary:
ALPC Port      : 1779
Desktop       : 123
Directory     : 254
EtwConsumer    : 6
EtwRegistration : 4661
Event         : 30287
File          : 7201
FilterCommunicationPort: 10
FilterConnectionPort: 2
IoCompletion   : 312
IoCompletionReserve: 1
Job           : 22
Key           : 4591
KeyedEvent    : 129
Mutant        : 2963
PcwObject     : 29
PowerRequest  : 6
Process       : 1019
Section       : 2991
Semaphore     : 4338
Session       : 18
SymbolicLink  : 5
Thread        : 3698
Timer         : 368
TmRm         : 20
TmTm         : 10
Token        : 613
TpWorkerFactory : 460
WindowStation : 224
WmiGuid       : 16
Total handles: 66156

c:\Tools>

```

8. При помощи *Handle.exe* можно закрывать дескрипторы объектов, открытых процессом, не завершая сам процесс. В качестве примера рассмотрим ситуацию, когда в приложении *Excel* открыт файл *test.xlsx*. Проверим дескрипторы открытых в системе файлов с

расширением *.xls\** с помощью команды *handle -a .xls*. В результате мы получим перечень из двух объектов типа *File*.

```

Администратор: Командная строка
c:\Tools>handle -a .xls

Handle v3.51
Copyright (C) 1997-2013 Mark Russinovich
Sysinternals - www.sysinternals.com

EXCEL.EXE pid: 8812 type: File BD8: D:\Users\avtimofeev\Documents\File.xlsx
EXCEL.EXE pid: 8812 type: File CD8: D:\Users\avtimofeev\Documents\~$File.xlsx
c:\Tools>

```

Закроем принудительно дескриптор одного из этих объектов с помощью команды

`handle -c CD8 -p 8812 -y`

```

Администратор: Командная строка - handle -c CD8 -p 8812 -y
c:\Tools>handle -c CD8 -p 8812 -y

Handle v3.51
Copyright (C) 1997-2013 Mark Russinovich
Sysinternals - www.sysinternals.com

CD8: File (R--) D:\Users\avtimofeev\Documents\~$File.xlsx
Handle closed.
c:\Tools>

```

Затем снова проверим дескрипторы открытых в системе файлов с расширением *.xls\** с помощью команды

`handle -a .xls`

```

Администратор: Командная строка
c:\Tools>handle -a .xls

Handle v3.51
Copyright (C) 1997-2013 Mark Russinovich
Sysinternals - www.sysinternals.com

EXCEL.EXE pid: 8812 type: File BD8: D:\Users\avtimofeev\Documents\File.xlsx
EXCEL.EXE pid: 8812 type: File CD8: D:\Users\avtimofeev\Documents\~$File.xlsx
c:\Tools>

```

Повторите подобный эксперимент и запишите в отчет результаты.

**Внимание:** Поскольку процесс, владеющий дескриптором объекта, не знает о том, что этот дескриптор закрыт, использование этой функции может привести к повреждению данных или сбою в приложении; закрытие дескриптора в системном процессе или критически важном процессе пользовательского режима, таком как *csrss*, может привести к сбою в системе. Кроме того, при последующем распределении ресурсов тем же процессом может быть получено старое значение дескриптора, так как оно больше не используется. В этом случае процесс может получить доступ не к тому объекту, который ожидает.

9. Подготовьте итоговый отчет с развернутыми выводами по заданию.

**Задание 1.2.** Изучить хранение информации об объектах процесса.

В рамках этого задания Вам будет предложено использовать программу *LiveKD*, которая позволяет запускать отладчик ядра Microsoft Kd, входящий в Windows Driver Kit (WDK). *LiveKD* позволяет выполнять просмотр моментального снимка работающей локальной системы, не перезагружая систему в режим отладки. Поскольку *LiveKd* получает дампы физической памяти, возможны ситуации, когда структуры данных находятся в процессе изменения системой и не являются согласованными. Запускаясь, отладчик каждый раз начинает работать со свежим снимком системы. Для его обновления выполните выход из отладчика (команда q), и затем ответьте утвердительно на вопрос о перезапуске *LiveKd*.

Для получения справки по той или иной команде отладчика, Вы можете просто набрать команду *.hh <имя команды>*, после чего Вам будет предложен для изучения соответствующий раздел файла справочной системы. Если вдруг Вам потребуется прервать работу некоторой команды, то Вы можете нажать Ctrl+Break, после этого Вам будет предложено перезапустить *LiveKD* или выйти из отладчика.

**Указания к выполнению.**

1. Загрузите пакет Windows Driver Kit (WDK), находящийся по адресу <http://msdn.microsoft.com/en-US/windows/hardware/hh852362>
2. Установите на компьютер пакет Windows Driver Kit (WDK) с настройками по умолчанию.
3. Загрузите программу *LiveKd*, размещенную по адресу <http://technet.microsoft.com/ru-ru/sysinternals/bb897415.aspx>.
4. Извлеките из архива утилиту *LiveKd.exe* и поместите в каталог **c:\Tools\LiveKD**.
5. Запустите командную строку от имени администратора, перейдите в каталог **c:\Tools\LiveKD** и запустите утилиту *LiveKd.exe*. В процессе запуска программы Вам могут быть заданы некоторые вопросы, касающиеся настроек запуска, отвечайте на них утвердительно.

```

c:\Tools\LiveKD>livekd.exe

LiveKd v5.3 - Execute kd/windbg on a live system
Sysinternals - www.sysinternals.com
Copyright (C) 2000-2012 Mark Russinovich and Ken Johnson

Symbols are not configured. Would you like LiveKd to set the _NT_SYMBOL_PATH
directory to reference the Microsoft symbol server so that symbols can be
obtained automatically? (y/n) y

Enter the folder to which symbols download (default is c:\symbols):
Launching C:\Program Files\Windows Kits\8.0\Debuggers\x86\kd.exe:
C
Microsoft (R) Windows Debugger Version 6.2.9200.20512 X86
Copyright (c) Microsoft Corporation. All rights reserved.

Loading Dump File [C:\Windows\livekd.dmp]
Kernel Complete Dump File: Full address space is available

Comment: 'LiveKD live system view'
Symbol search path is: srv*c:\Symbols*http://msdl.microsoft.com/download/symbols

Executable search path is:
Windows 7 Kernel Version 7601 (Service Pack 1) MP (2 procs) Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS

```

6. С помощью команды *!handle* можно получить информацию об открытых дескрипторах объектов. Команда имеет следующую форму:

*!handle* <индекс\_дескриптора> <флаги> <идентификатор\_процесса>.

Индекс дескриптора определяет элемент в таблице дескрипторов (0 – вывод всех дескрипторов). Вы можете указывать флаги, являющиеся битовыми масками, где бит 0 означает, что нужно вывести лишь информацию из элемента таблицы, бит 1 – показать не только используемые, но и свободные дескрипторы, а бит 2 – сообщить информацию об объекте, на который ссылается дескриптор. Следующая команда выводит полную информацию о таблице дескрипторов в процессе с идентификатором *0x1098* (4248<sub>10</sub>).

```

0: kd> !handle 0 3 1098

Searching for Process with Cid == 1098
PROCESS 866aa7d8 SessionId: 1 Cid: 1098 Peb: 7ffd3000 ParentCid: 162c
DirBase: bd59f300 ObjectTable: cc202c30 HandleCount: 188.
Image: calc.exe

Handle table at cc202c30 with 188 entries in use

0004: Object: 918f67d0 GrantedAccess: 00000003 Entry: 9cf99008
Object: 918f67d0 Type: (86106610) Directory
ObjectHeader: 918f67b8 (new version)
HandleCount: 107 PointerCount: 145
Directory Object: 8d405ed0 Name: KnownDlls

Hash Address Type Name
----
00 9a386540 Section IMAGEHLP.dll
9a3aac10 Section gdi32.dll
9de08d58 Section kernelbase.dll
02 9a395fd8 Section NORMALIZ.dll
03 9a3a5ee0 Section ole32.dll
9a37adb8 Section URLMON.dll
04 9a378040 Section USP10.dll
05 9a3f65f8 Section DEVOBJ.dll
06 9a3aec00 Section SHELL32.dll
9a3f6498 Section CFGMGR32.dll
9a3b1b18 Section WLDAP32.dll
09 9a381fd8 Section user32.dll
14 9a3f35a0 Section MSASN1.dll
16 918e4158 SymbolicLink KnownDllPath
9a3fe110 Section COMCTL32.dll
17 9a394cf8 Section PSAPI.DLL
9a3f4e08 Section CRYPT32.dll
18 9a378a28 Section advapi32.dll
9a3d1b50 Section OLEAUT32.dll
19 9a3d1590 Section SHLWAPI.dll
9a3957f0 Section IERTUTIL.dll
919a7da0 Section ntdll.dll
20 9a3dfd10 Section WS2_32.dll
21 9a37ae40 Section LPK.dll
22 9a3a5f70 Section sechost.dll
23 9a386870 Section COMDLG32.dll
24 9a3b4750

```



Команда *!handle 284 3 1098* позволяет получить подробную информацию об выбранном объекте (*0x284*) заданного процесса (*0x1098*).

```

0: kd> !handle 284 3 1098
Searching for Process with Cid == 1098
PROCESS 866aa7d8 SessionId: 1 Cid: 1098 Peb: 7fffd3000 ParentCid: 162c
DirBase: bd59f300 ObjectTable: cc202c30 HandleCount: 188.
Image: calc.exe

Handle table at cc202c30 with 188 entries in use
0284: Object: 86457828 GrantedAccess: 001fffff Entry: 9cf99508
Object: 86457828 Type: (8610bc18) Thread
ObjectHeader: 86457810 (new version)
HandleCount: 2 PointerCount: 4

0: kd>

```

Команда *!handle 284 3 1098* позволяет получить подробную информацию об выбранном объекте (*0x284*) заданного процесса (*0x1098*).

Выполните с помощью команды *!handle* вывод списка объектов, принадлежащих процессу, выбранному в пункте 3 прошлого задания. Запишите в отчет результаты выполнения данного шага, сравните полученные сведения с данными из пункта 3 прошлого задания.

7. С помощью команды *!process* можно получить информацию о всех процессах вычислительной системы. Команда *!process 1098 0* позволяет получить краткую информацию о заданном процессе (*0x1098*), в том числе получить адрес его таблицы объектов (ObjectTable).

```

0: kd> !process 1098 0
Searching for Process with Cid == 1098
Cid handle table at 8d401098 with 2359 entries in use

PROCESS 866aa7d8 SessionId: 1 Cid: 1098 Peb: 7fffd3000 ParentCid: 162c
DirBase: bd59f300 ObjectTable: cc202c30 HandleCount: 188.
Image: calc.exe

0: kd>

```

Получите с помощью команды *!process* информацию о процессе, выбранном в пункте 3 прошлого задания. Запишите в отчет результаты выполнения данного шага.

8. Чтобы проанализировать поля таблицы объектов процесса, следует воспользоваться командой *dt\_handle\_table <адрес\_таблицы>*.

```

0: kd> dt _handle_table cc202c30
ntdll!_HANDLE_TABLE
+0x000 TableCode      : 0x9cf99000
+0x004 QuotaProcess   : 0x866aa7d8 _EPROCESS
+0x008 UniqueProcessId : 0x00001098 Void
+0x00c HandleLock     : _EX_PUSH_LOCK
+0x010 HandleTableList : _LIST_ENTRY [ 0xcf5bf0c8 - 0xce1040a0 ]
+0x018 HandleContentionEvent : _EX_PUSH_LOCK
+0x01c DebugInfo      : (null)
+0x020 ExtraInfoPages : 0n0
+0x024 Flags          : 0
+0x028 StrictFIFO     : 0y0
+0x02c FirstFreeHandle : 0x2f0
+0x030 LastFreeHandleEntry : 0x9cf99ff8 _HANDLE_TABLE_ENTRY
+0x034 HandleCount    : 0xbc
+0x038 NextHandleNeedingPool : 0x800
+0x03c HandleCountHighWatermark : 0xbd
0: kd>

```

Для дальнейшего анализа таблицы объектов наибольший интерес представляет поле *TableCode*. Поле *TableCode* структуры *HANDLE\_TABLE* в двух младших битах *Attr* содержит данные о числе уровней таблицы (от одного до трех), остальные биты *TableCode* представляют собой указатель на таблицу первого уровня.

Например, если *Attr* равен нулю, то число таблиц равно единице, следовательно, первый уровень содержит элементы *TABLE\_ENTRY* (то есть *TableCode* указывает на массив *TABLE\_ENTRY*). Если *Attr* равен единице, то таблиц две, и значит, *TableCode* указывает на промежуточную (вторую) таблицу, каждый элемент которой и включает указатели на *TABLE\_ENTRY* и т.д.

Получите с помощью команды *dt \_handle\_table* информацию о таблице объектов выбранного ранее процесса. Запишите в отчет результаты выполнения данного шага.

9. Проанализируем с помощью команды *dd <физический\_адрес>* первые 64 16-разрядных слова, размещенные по адресу одноуровневой таблицы объектов процесса. Обратите внимание, что каждому объекту ставится в соответствие четыре 16-разрядных слова. В качестве идентификатора объекта используется смещение элемента объекта в 16-разрядных словах относительно начала таблицы. Элемент 0 не используется для хранения объекта. Поэтому первый объект в таблице имеет идентификатор 4, второй – 8 и т.д.

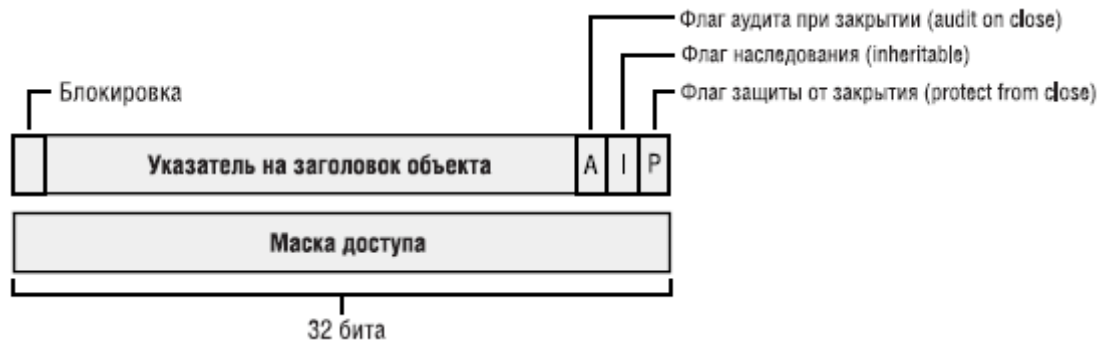
```

0: kd> dd 9cf99000
9cf99000 00000000 ffffffff 918f67b9 00000003
9cf99010 868eb359 00100020 8666e2e1 00100020
9cf99020 862f89e9 00100020 885da909 001f0001
9cf99030 c553d741 00020019 c87c9e79 00000001
9cf99040 8bb24fb9 00000804 885bf241 021f0003
9cf99050 885fa5a9 000f037f 885dc759 000f01ff
9cf99060 885fa5a9 000f037f cbfc3761 000f003f
9cf99070 86c41021 00000804 8b93b661 00000804
0: kd>

```

Структура элемента таблицы дескрипторов представлена на рисунке ниже.





Чтобы получить адрес заголовка объекта необходимо выполнить следующую операцию над младшими 32 битами элемента таблицы дескрипторов:  $(\text{<младшие 32 бита>} \mid 0x80000000) \& 0xffffffff8$ . Так в рассмотренном выше примере в таблице объектов под дескриптором 4 хранится значение 918F67B9 00000003, в этом случае адрес заголовка объекта будет:  $0x918F67B9 \mid 0x80000000) \& 0xffffffff8 = 0x918F67B8$ .

Получите с помощью команды *dd* содержимое первых записей таблице объектов выбранного ранее процесса, определите физические адреса объектов с дескрипторами 4, 8, 12. Запишите в отчет результаты выполнения данного шага.

10. Для чтения заголовка объекта по его адресу необходимо воспользоваться командой *dt \_object\_header <физический\_адрес>*. Например, ниже приведен заголовок объекта с дескриптором 4 из нашего примера.

```

Администратор: Командная строка - LiveKD\livekd.exe
0: kd> dt _object_header 918F67B8
nt!_OBJECT_HEADER
+0x000 PointerCount      : 0n144
+0x004 HandleCount      : 0n106
+0x004 NextToFree       : 0x0000006a Void
+0x008 Lock              : _EX_PUSH_LOCK
+0x00c TypeIndex        : 0x3 ''
+0x00d TraceFlags       : 0 ''
+0x00e InfoMask         : 0xa ''
+0x00f Flags            : 0x10 ''
+0x010 ObjectCreateInfo : 0x83379a00 _OBJECT_CREATE_INFORMATION
+0x010 QuotaBlockCharged : 0x83379a00 Void
+0x014 SecurityDescriptor : 0x9a3758ba Void
+0x018 Body             : _QUAD
0: kd>

```

Получите с помощью команды *dt \_object\_header* содержимое заголовков объектов с дескрипторами 4, 8, 12. Запишите в отчет результаты выполнения данного шага.

11. Чтобы получить описание самого объекта необходимо обратиться к полю *Body* заголовка объекта по смещению 0x18. Для этого необходимо использовать команду *!object*, пример приведен ниже.

```

0: kd> !object 918f67b8+18
Object: 918f67d0 Type: (86106610) Directory
ObjectHeader: 918f67b8 (new version)
HandleCount: 106 PointerCount: 144
Directory Object: 8d405ed0 Name: KnownDlls

Hash Address Type Name
----
00 9a386540 Section IMAGEHLP.dll
9a3aac10 Section gdi32.dll
9de08d58 Section kernelbase.dll
02 9a395fd8 Section NORMALIZ.dll
03 9a3a5ee0 Section ole32.dll
9a37adb8 Section URLMON.dll
04 9a378040 Section USP10.dll
05 9a3f65f8 Section DEVOBJ.dll
06 9a3aec00 Section SHELL32.dll
9a3b1b18 Section WLDAP32.dll
9a3f6498 Section CFGMGR32.dll
09 9a381fd8 Section user32.dll
14 9a3f35a0 Section MSASN1.dll
16 918e4158 SymbolicLink KnownDllPath
9a3fe110 Section COMCTL32.dll
17 9a394cf8 Section PSAPI.dll
9a3f4e08 Section CRYPT32.dll
18 9a378a28 Section advapi32.dll
9a3d1b50 Section OLEAUT32.dll
19 9a3d1590 Section SHLWAPI.dll
9a3957f0 Section IERTUTIL.dll
919a7da0 Section ntdll.dll
20 9a3dfd10 Section WS2_32.dll
21 9a37ae40 Section LPK.dll
22 9a3a5f70 Section sechost.dll
23 9a386870 Section COMDLG32.dll
24 9a3b4750 Section difxapi.dll
25 9a3f8e78 Section Setupapi.dll
26 9a378178 Section MSCTF.dll
9a38e890 Section WININET.dll
27 9a37ed38 Section IMM32.dll
9a3f4ef0 Section WINTRUST.dll
28 9a3bbc58 Section MSVCRT.dll
31 9a3a5fd8 Section rpcrt4.dll
9a3b1fd8 Section c1bcatq.dll
32 9a3c3518 Section kernel32.dll
35 9a3f3470 Section NSI.dll
0: kd>

```

Получите с помощью команды *!object* описание объектов с дескрипторами 4, 8, 12. Запишите в отчет результаты выполнения данного шага.

12. Для получения сведений о типе объекта необходимо использовать команду *dt \_object\_type*, в качестве параметра которой передается значение поля *Type*, полученное после использования команды *!object*. Структура типа включает имя типа объекта, счетчики активных объектов этого типа, а также счетчики пикового числа дескрипторов и объектов данного типа. В поле *TypeInfo* хранится указатель на структуру данных, в которой содержатся атрибуты, общие для всех объектов этого типа, а также указатели на методы типа.

```

0: kd> dt _object_type 86106610
ntdll!_OBJECT_TYPE
+0x000 TypeList : _LIST_ENTRY [ 0x86106610 - 0x86106610 ]
+0x008 Name : _UNICODE_STRING "Directory"
+0x010 DefaultObject : 0x83386860 Void
+0x014 Index : 0x3
+0x018 TotalNumberOfObjects : 0x3a
+0x01c TotalNumberOfHandles : 0xee
+0x020 HighWaterNumberOfObjects : 0x40
+0x024 HighWaterNumberOfHandles : 0x133
+0x028 TypeInfo : _OBJECT_TYPE_INITIALIZER
+0x078 TypeLock : _EX_PUSH_LOCK
+0x07c Key : 0x65726944
+0x080 CallbackList : _LIST_ENTRY [ 0x86106690 - 0x86106690 ]
0: kd>

```

Получите с помощью команды *dt \_object\_type* сведений о типах для объектов с дескрипторами 4, 8, 12. Запишите в отчет результаты выполнения данного шага.

13. Выполним анализ структуры данных, в которой содержатся атрибуты, общие для всех объектов типа, а также указатели на методы типа. Для вывода содержимого поля используем команду `dt _object_type_initializer <физический адрес>`, не забываем, что поле `TypeInfo` имеет смещение `0x28` относительно начала структуры сведений о типе объекта.

```

0: kd> dt _object_type_initializer 86106610+28
ntdll!_OBJECT_TYPE_INITIALIZER
+0x000 Length : 0x50
+0x002 ObjectTypeInfoFlags : 0xd
+0x002 CaseInsensitive : 0y1
+0x002 UnnamedObjectsOnly : 0y0
+0x002 UseDefaultObject : 0y1
+0x002 SecurityRequired : 0y1
+0x002 MaintainHandleCount : 0y0
+0x002 MaintainTypeList : 0y0
+0x002 SupportsObjectCallbacks : 0y0
+0x002 CacheAligned : 0y0
+0x004 ObjectTypeCode : 0
+0x008 InvalidAttributes : 0x100
+0x00c GenericMapping : _GENERIC_MAPPING
+0x01c ValidAccessMask : 0xf000f
+0x020 RetainAccess : 0
+0x024 PoolType : 1 ( PagedPool )
+0x028 DefaultPagedPoolCharge : 0x30
+0x02c DefaultNonPagedPoolCharge : 0xa8
+0x030 DumpProcedure : (null)
+0x034 OpenProcedure : (null)
+0x038 CloseProcedure : 0x834aea76 void nt!ObpCloseDirectoryObject+0
+0x03c DeleteProcedure : (null)
+0x040 ParseProcedure : (null)
+0x044 SecurityProcedure : 0x834a97d6 long nt!SeDefaultObjectMethod+0
+0x048 QueryNameProcedure : (null)
+0x04c OkayToCloseProcedure : (null)
0: kd>

```

Получите с помощью команды `dt _object_type_initializer` общих атрибутов и методов типов для объектов с дескрипторами 4, 8, 12. Запишите в отчет результаты выполнения данного шага.

14. Подготовьте итоговый отчет с развернутыми выводами по заданию.