

# Лабораторная работа №6

---

«ИЗУЧЕНИЕ ТЕХНИКИ ОТЛАДКИ ДРАЙВЕРОВ»

# Введение

---

Драйвер, строго говоря, – это фрагмент кода операционной системы, который позволяет ей обращаться к аппаратуре. Иными словами, драйвер представляет собой в принципе обычную программу, но работающую в режиме ядра или на «нулевом кольце защиты», известном из курса языков ассемблера.

При разработке драйвера или исследовании стороннего, уже написанного, возникает резонный вопрос об отладке. Отладка кода режима ядра существенно сложнее, чем отладка обычных программ. Так, если для отладки пользовательского кода можно использовать привычные дебаггеры, такие как Visual Studio, IDA, OllyDbg и т.п., то для отладки кода режима ядра требуются спецсредства.

# Постановка задачи

---

1. Изучение инструментов, предназначенных для отладки драйверов и их настроек
2. Изучение техники отладки драйверов
3. Применение способа отладки драйвера на практике

# Общие сведения

---

Обычно выделяют два подхода к отладке драйверов:

1. Отладчик работает на той же машине, что и исследуемый код
2. Отладчик работает на другой машине

При отладке драйверов и компонентов ОС в большинстве случаев используется второй способ, что обусловлено следующими причинами:

1. Практически при любой ошибке кода, выполняющегося на уровне ядра, система выдает сообщение о критической системной ошибке – так называемый “синий экран смерти, BSoD” и прекращает работу вместе с отладчиком.
2. При попадании на точку останова (int 3) система останавливается вместе с локальным отладчиком.

Существуют способы отладки драйверов на реальных машинах, к которым отладчик подключается по сети, COM-порту или по USB, но такой способ почти не практикуется. Обычно тестируемый код запускается на виртуальной машине, а отладчик подключается по именованному каналу.

В связи с вышеописанными особенностями, сформулируем начальные условия выполнения задания.

# Начальные условия

---

1. VMWare Workstation для запуска виртуальной машины
2. Виртуальная машина Windows XP SP3
3. Виртуальная машина Windows 7
4. Отладчик WinDBG , который входит в пакет WDK, установленный на виртуальной машине с Windows 7 (C:\WinDDK). В данной лабораторной работе используется версия 7 пакета, которую бесплатно можно скачать с сайта Microsoft.

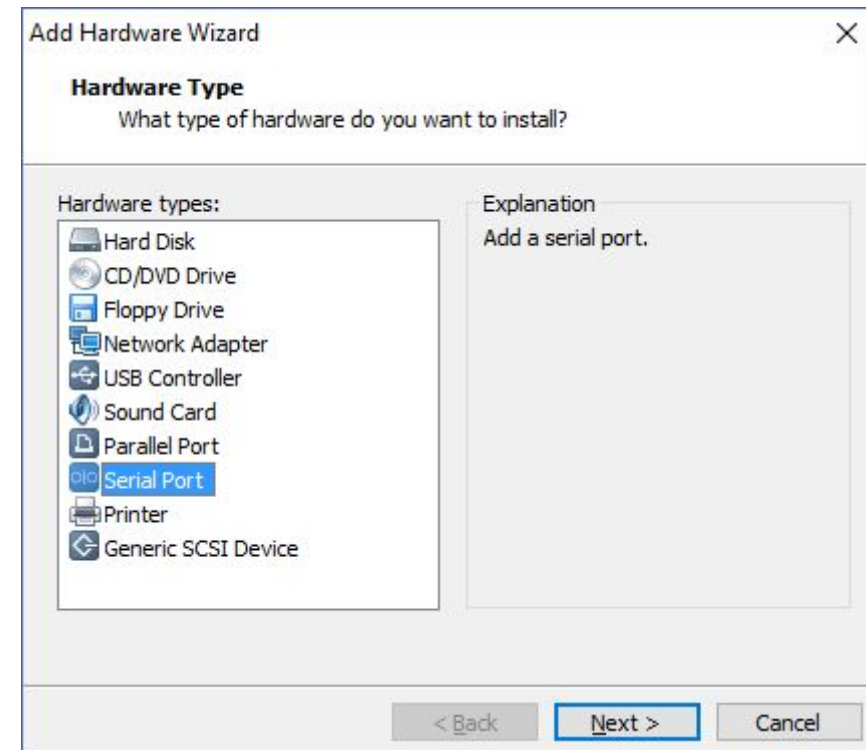
После того, как все необходимое установлено, можно переходить к выполнению лабораторной работы.

# Шаг 1.1 Настройка целевой системы Windows XP

---

Для настройки целевой системы, на которой будет производиться запуск драйвера (в данной лабораторной – это машина с ОС Windows XP) выполняем следующую последовательность действий.

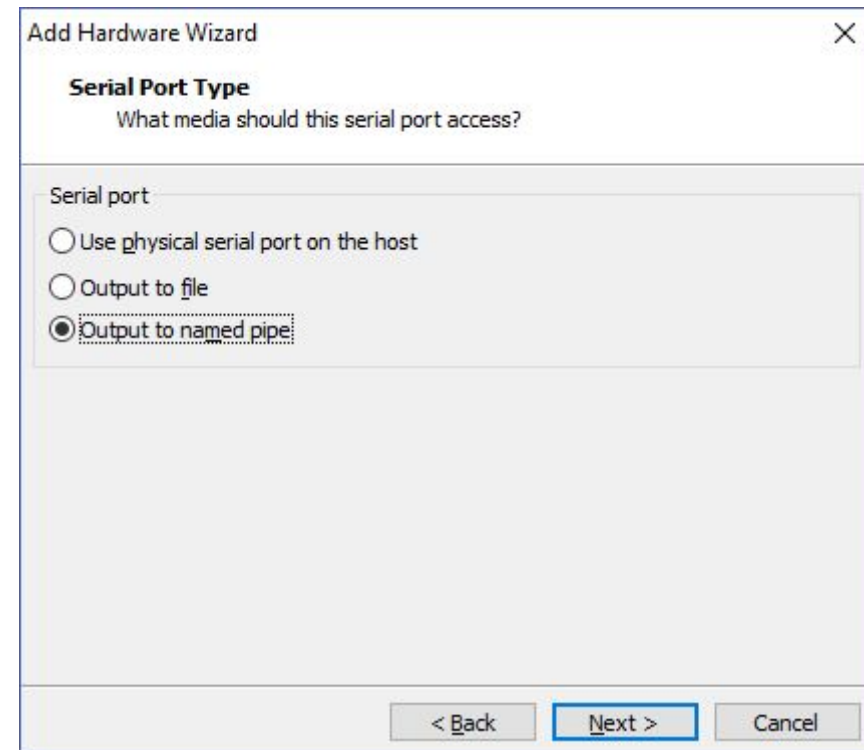
В пункте «Settings» вкладки «VM» выбираем вкладку «Hardware» и жмем кнопку «Add». В списке «Hardware type» выбираем «Serial Port», нажимаем кнопку «Next».



# Шаг 1.1 Настройка целевой системы Windows XP

---

Выбираем тип последовательного порта «Output to named pipe», нажимаем «Next»

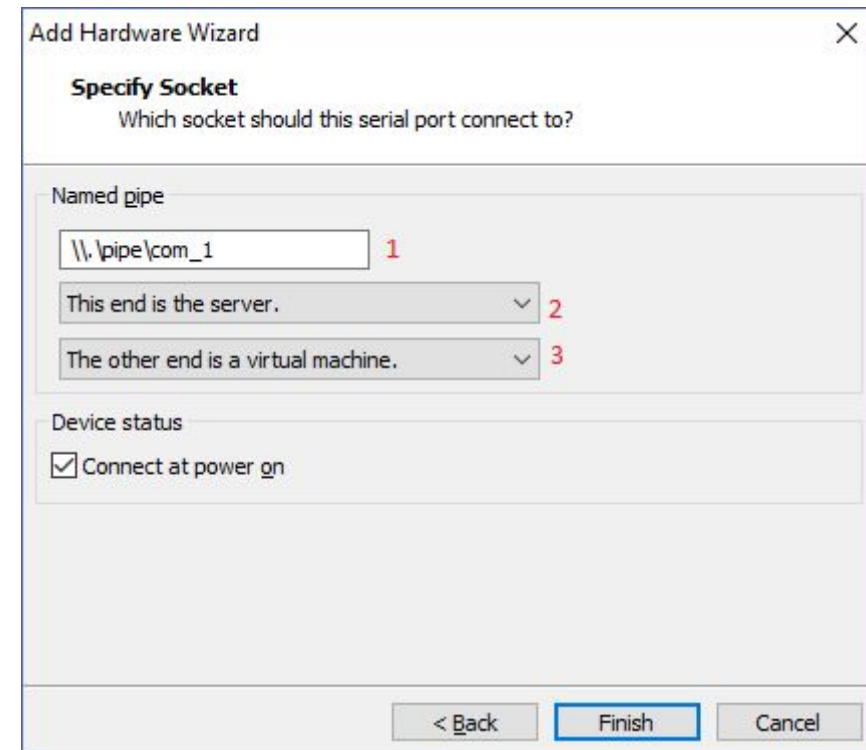


# Шаг 1.1 Настройка целевой системы Windows XP

1. Имя pipe. Можно оставить по умолчанию. Если виртуальных машин для отладки много (например, различные версии ОС), то есть смысл дать каналам осмысленные имена.
2. Выбираем «This end is the server»
3. Выбираем «The other end is virtual machine»

Устанавливаем значение параметра «Connect at power on»

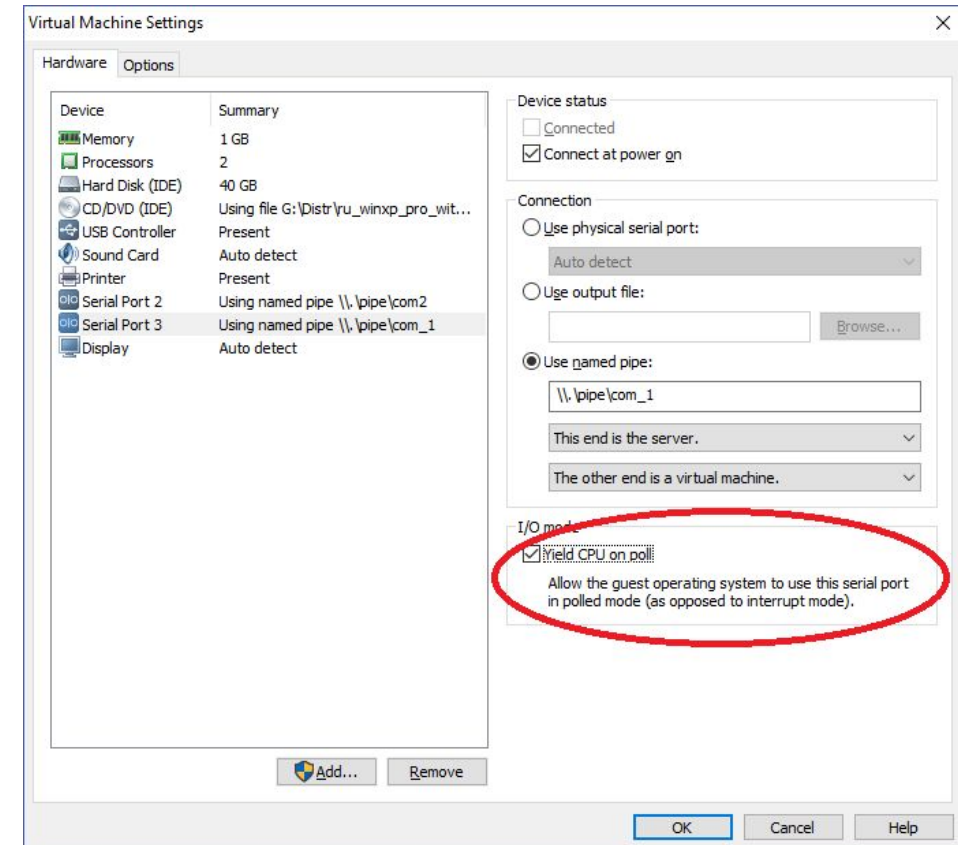
Нажимаем «Finish».





# Шаг 1.1 Настройка целевой системы Windows XP

После того, как выполнены предыдущие шаги и порт создан, на вкладке параметров порта установим параметр «Yield CPU on poll» и нажмём «ОК»



# Шаг 1.1 Настройка целевой системы Windows XP

---

Теперь нужно настроить непосредственно операционную систему, поэтому включим машину с ОС Windows XP.

Включим режим отладки - для этого нужно дописать несколько строк в файл **Boot.ini**. Чтобы его отредактировать - заходим в «Мой компьютер», далее «Свойства» - вкладка «Дополнительно». «Загрузка и восстановление» - «Параметры» - «Правка». Файл Boot.ini открыт.

Копируем строку, с помощью которой загружается система, вставляем ее и дописываем к ней следующее содержание: /noexecute=optin /fastdetect /sos /bootlog /debug /debugport=<имя\_порта\_из\_настройки\_виртуальной\_машины> /baudrate=115200.

Например, содержимое файла может выглядеть как на скриншоте справа.

Отметим, что данный способ работает только на ОС не новее Windows XP. Если Вы используете виртуальную машину с более новой ОС воспользуйтесь утилитой bcdedit.

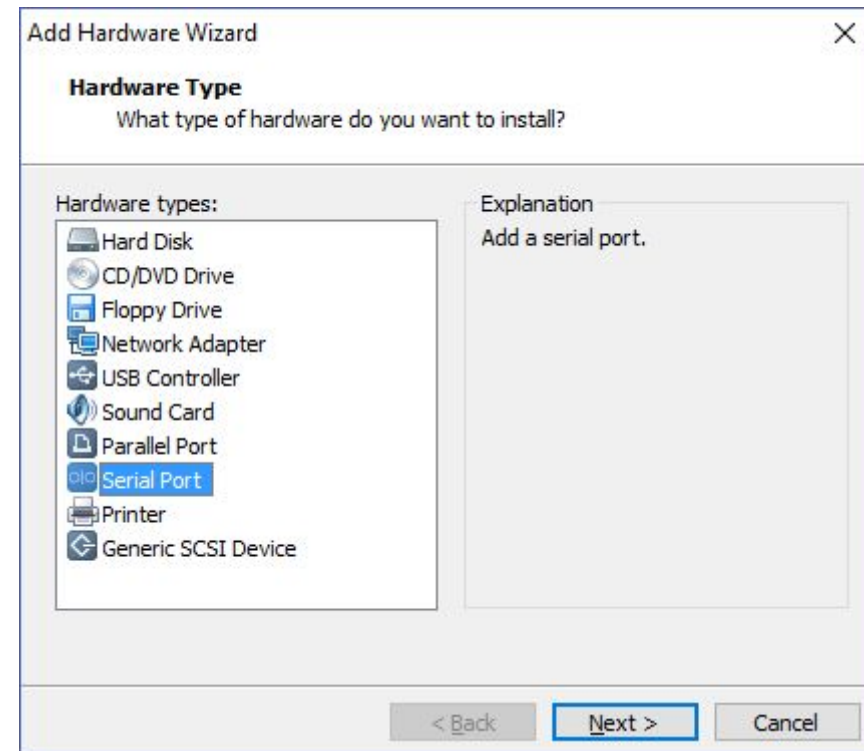
После изменений содержимого файла boot.ini, систему необходимо перезагрузить.

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP
Professional - DEBUGGED" /noexecute=optin /fastdetect /sos
/bootlog /debug /debugport=com_1 /baudrate=115200
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP
Professional" /noexecute=optin /fastdetect
```

# Шаг 1.2 Настройка целевой системы Windows 7

Для настройки целевой системы, на которой будет производиться отладка драйвера (в данной лабораторной – это машина с ОС Windows 7) выполняем следующую последовательность действий.

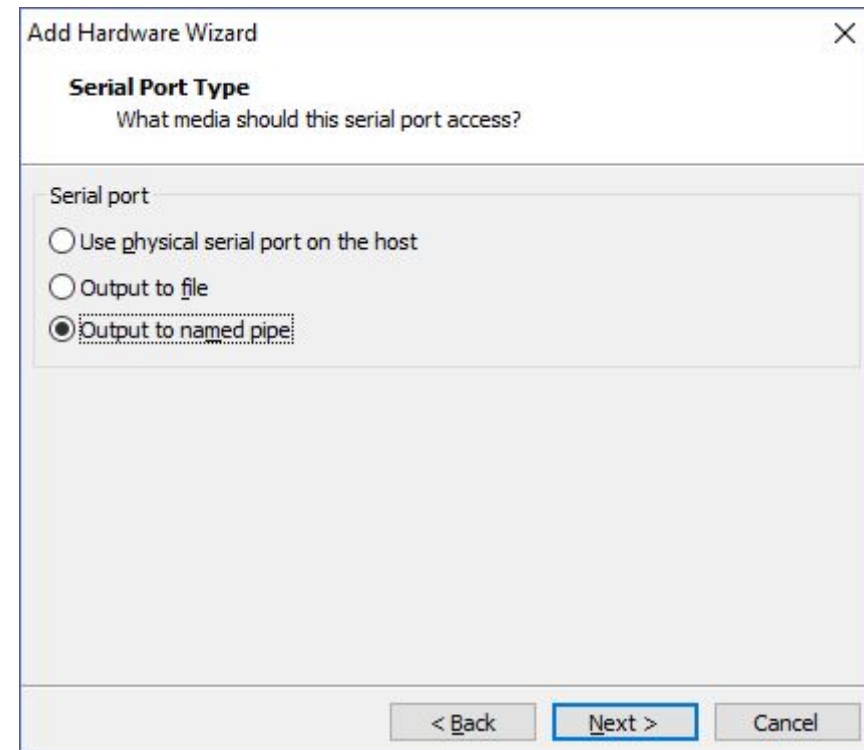
В пункте «Settings» вкладки «VM» выбираем вкладку «Hardware» и жмем кнопку «Add». В списке «Hardware type» выбираем «Serial Port», нажимаем кнопку «Next».



# Шаг 1.2 Настройка целевой системы Windows 7

---

Выбираем тип последовательного порта «Output to named pipe», нажимаем «Next»



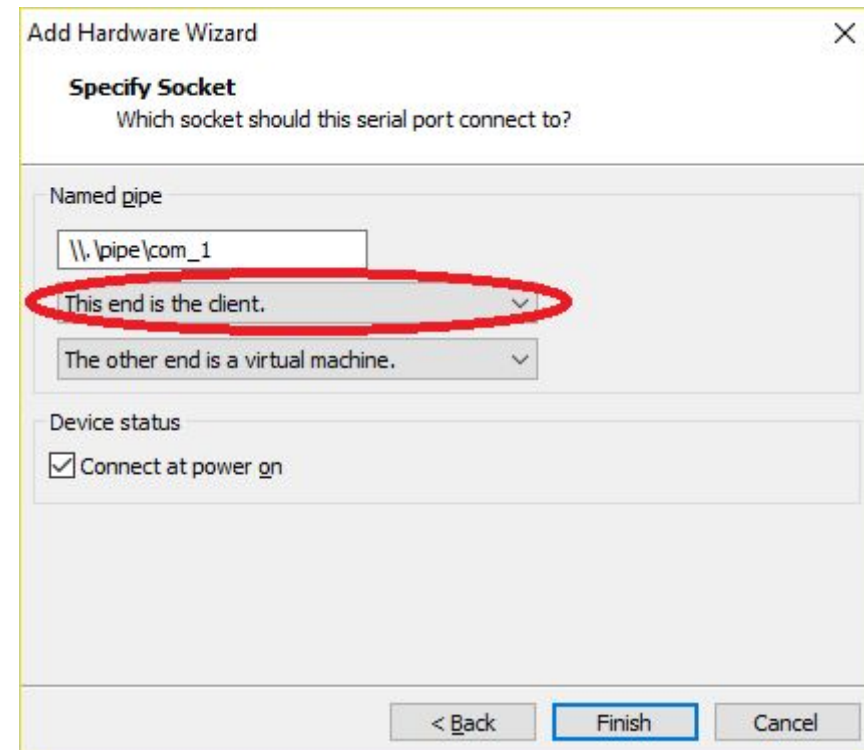
# Шаг 1.2 Настройка целевой системы Windows 7

1. Имя pipe. **Должно совпадать с именем пайпа из настройки Windows XP.**
2. Выбираем «This end is the **client**». Обратите внимание, что в отличие от настройки Windows XP, устанавливается параметр клиента
3. Выбираем «The other end is virtual machine»

Устанавливаем значение параметра «Connect at power on»

Нажимаем «Finish».

Более, порт не настраивается и, в отличие от настройки Windows XP, параметр «Yield CPU on poll» **не** устанавливается.



# Шаг 2. Настройка отладчика

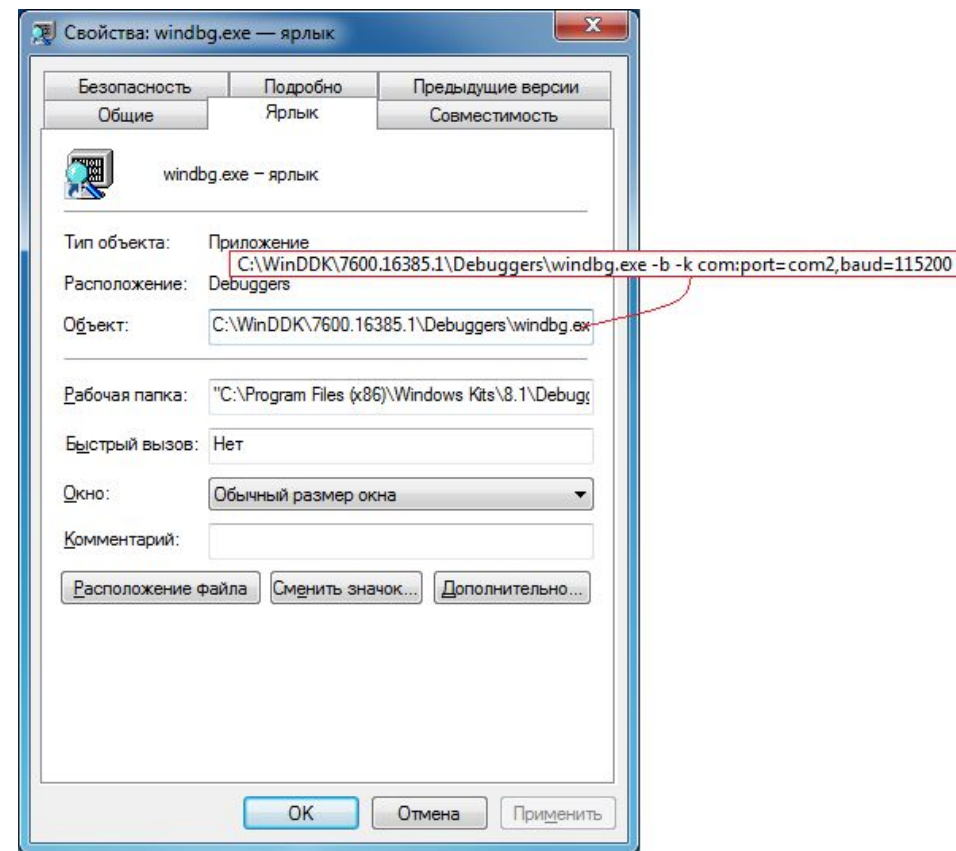
Произведем настройку WinDbg, которая ускорит запуск отладчика.

Для отладки системных компонентов нужна символьная информация. Для того, чтобы WinDbg смог использовать уже установленные символы, нужно либо каждый раз указывать путь, либо добавить эту информацию в параметры запуска. Чтобы подключиться к конкретному pipe, необходимо указать его имя. Также, как и в случае с символами, имя указывается либо после запуска, либо как ключ.

Удобно для каждого имени pipe создать свой ярлык с параметрами, что сэкономит время при каждом запуске. Итак, в рабочей директории создаем ярлык со следующими параметрами:

*Объект: "Путь\_до\_windbg.exe" -b -k com:port=ИМЯ\_pipe,baud=115200*

Например, это может выглядеть как на скриншоте.

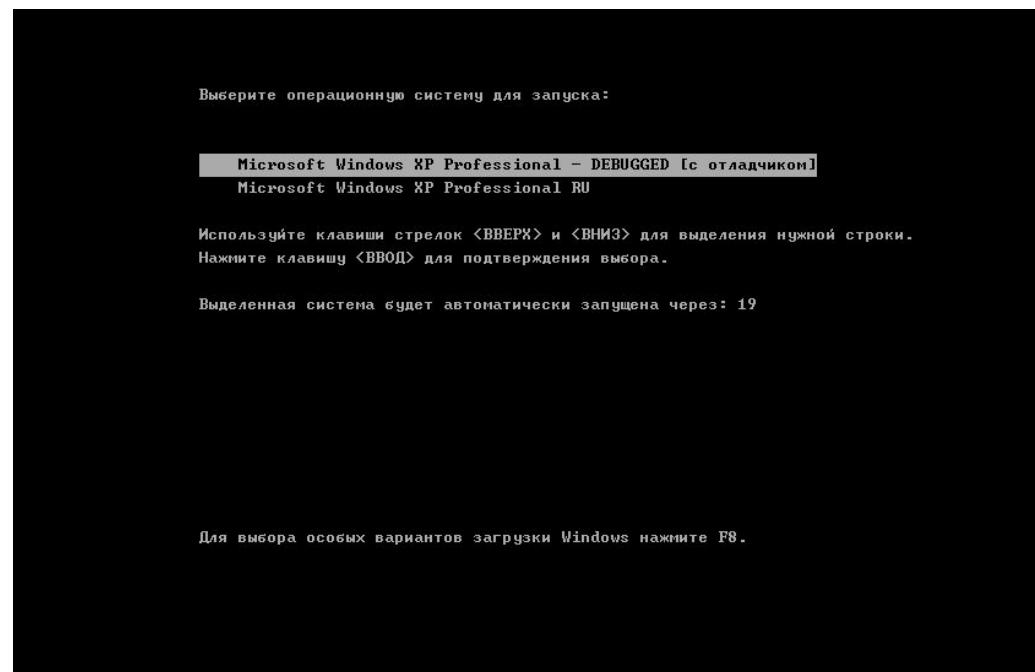


# Шаг 3. Проверка настроек

---

Для проверки запустим виртуальные машины.

При запуске Windows XP выбираем пункт меню загрузки с отладчиком.



# Шаг 3. Проверка настроек

---

В момент загрузки При появлении логотипа Windows запускаем WinDbg на машине с ОС Windows 7 с помощью созданного ранее ярлыка. Если все сделано верно, то на экране, в окне «Command» будет то же, что и на скриншоте.

```
Microsoft (R) Windows Debugger Version 6.12.0002.633 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

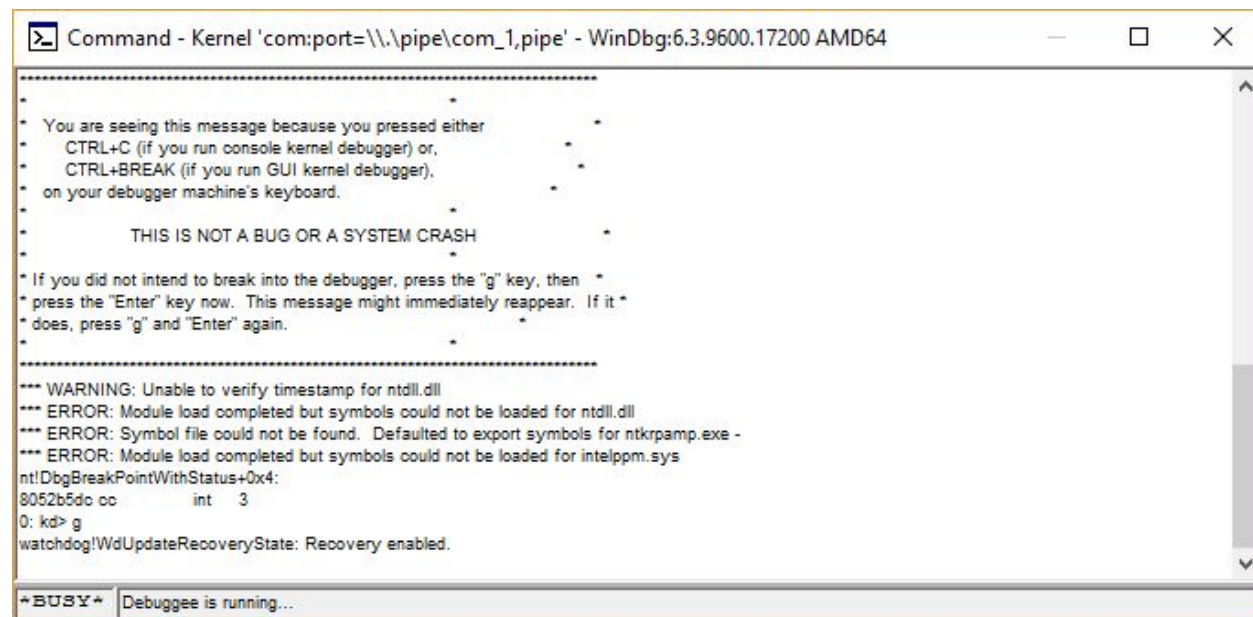
Opened \\.\com2
Waiting to reconnect...
Connected to Windows XP 2600 x86 compatible target at (Mon Dec 12 22:11:15.413 2016 (UTC + 3:00)), ptr64 FALSE
Kernel Debugger connection established. (Initial Breakpoint requested)
Symbol search path is: *** Invalid ***
*****
* Symbol loading may be unreliable without a symbol search path. *
* Use .symfix to have the debugger choose a symbol path. *
* After setting your symbol path, use .reload to refresh symbol locations. *
*****
Executable search path is:
*****
* Symbols can not be loaded because symbol path is not initialized. *
*
* The Symbol Path can be set by: *
* using the _NT_SYMBOL_PATH environment variable. *
* using the -y <symbol_path> argument when starting the debugger. *
* using .sympath and .sympath+ *
*****
*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntkrpxmp.exe -
Windows XP Kernel Version 2600 (Service Pack 3) MP (2 procs) Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 2600.xpsp.080413-2111
Machine Name:
```



# Шаг 3. Проверка настроек

Если на предыдущем шаге что-то не получилось и ниже надписи “Waiting to reconnect” ничего нет, значит отладчик не подключился. Возможно, Вы просто не успели.

В качестве возможного решения, предлагается приостановить систему, нажав комбинацию клавиш «*Ctrl+Break*», и, если в окне “Command” появилось то же, что и на скриншоте, то все в порядке. Возобновите работу системы, нажав «*f5*», и продолжайте работать.



```
Command - Kernel 'com:port=\\.\pipe\com_1,pipe' - WinDbg:6.3.9600.17200 AMD64
-----
* You are seeing this message because you pressed either
* CTRL+C (if you run console kernel debugger) or,
* CTRL+BREAK (if you run GUI kernel debugger),
* on your debugger machine's keyboard.
*
* THIS IS NOT A BUG OR A SYSTEM CRASH
*
* If you did not intend to break into the debugger, press the "g" key, then
* press the "Enter" key now. This message might immediately reappear. If it
* does, press "g" and "Enter" again.
*
-----
*** WARNING: Unable to verify timestamp for ntdll.dll
*** ERROR: Module load completed but symbols could not be loaded for ntdll.dll
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for ntkrnlmp.exe -
*** ERROR: Module load completed but symbols could not be loaded for intelppm.sys
nt!DbgBreakPoint(WithStatus+0x4:
8052b5dc cc      int     3
0: kd> g
watchdog!WdUpdateRecoveryState: Recovery enabled.

*BUSY* | Debuggee is running...
```

# Шаг 4. Написание тестового драйвера

---

Для того, чтобы написанный в среде Visual studio драйвер корректно запускался на гостевой машине, необходимо произвести настройки проекта.

Написание драйвера начинается с создания проекта Win32 и его настройки. Значения параметров для разделов VC++ Directories и C/C++ приведены на слайде.

1) TargetExtension = .sys

2) VC++ Directories:

Include Directories: добавить путь до WDK\inc\ddk  
(C:\WinDDK\7600.16385.1\inc\ddk)

Library Directories: добавить путь до WDK\lib\wpx\i386  
(C:\WinDDK\7600.16385.1\lib\wpx\i386)

3) C/C++:

Debug Information Format = 'C7 compatible (/Z7)'

Optimization = 'Disabled'

Enable Intrinsic Function = 'Yes'

Preprocessor Definitions = '\_DEBUG(или NDEBUG, если в Release);\_X86\_=1;i386=1;STD\_CALL;NT\_UP=1;CONDITION\_HANDLING;WINNT=1;WIN32\_LEAN\_AND\_MEAN=1;DEVL=1;FPO=1;\_DLL=1;\_IDWBUILD;WIN32=100;NT1X=100;NTUP=1;WIN32;\_WINDOWS;UNICODE;try=\_\_try;except=\_\_except;wcsicmp=\_wcsicmp;%(PreprocessorDefinitions)'

EnableString Pooling = 'Yes'

Security Check = 'Disable Security Check'

Enable Function-Level Linking = 'Yes'

Calling Convention = '\_stdcall'

Additional Options = '/D /Zel %(AdditionalOptions)'

Basic Runtime Check = 'Default'

# Шаг 4. Написание тестового драйвера

---

Далее производится настройка линковщика. Необходимые параметры указаны на слайде.

Linker:

```
Additional Dependencies =  
'ntoskrnl.lib;int64.lib;hal.lib;ndis.lib;%(AdditionalD  
ependencies)'
```

```
Additional Options = '/DRIVER /Subsystem:Native  
/FULLBUILD /FORCE:MULTIPLE /OPT:"REF"  
%(AdditionalOptions)'
```

```
Generate Manifest = 'No'  
Ignore All Default Libraries = 'Yes'  
Image Has Safe Exception Handlers = 'Yes'  
Merge Sections = '.CRT=.rdata'  
SubSystem = ''  
Entry Point = DriverEntry  
Target Machine = MachineX86 (/MACHINE:X86)
```

# Шаг 4. Написание тестового драйвера

---

В настроенный проект добавим файл “HelloWorldDriver.c”, содержащий точку входа DriverEntry и приветственное сообщение, как это сделано справа.

Компилируем. Если все прошло нормально, то в папке bin/Debug появится файл Driver.sys – это и есть файл драйвера.

```
1 #include <ntddk.h>
2 NTSTATUS NTAPI DriverEntry(PDRIVER_OBJECT driverObject, PUNICODE_STRING
3 registryPath)
4 {
5     UNREFERENCED_PARAMETER(driverObject);
6     UNREFERENCED_PARAMETER(registryPath);
7
8     DbgPrint("Hello, world!!!");
9
10    return STATUS_SUCCESS;
11 }
```

# Шаг 5. Проверка работы драйвера

Запускаем WinDbg через созданный ранее ярлык, запускаем виртуальную машину. На заставке с выбором ОС выбираем вариант с припиской “[с отладчиком]”. Ожидаем загрузки.

На диске C: создадим папку Driver, куда скопируем получившийся Driver.sys. Запускаем командную строку и выполняем команды:

```
sc create Driver binPath= C:/Driver/Driver.sys type= kernel
```

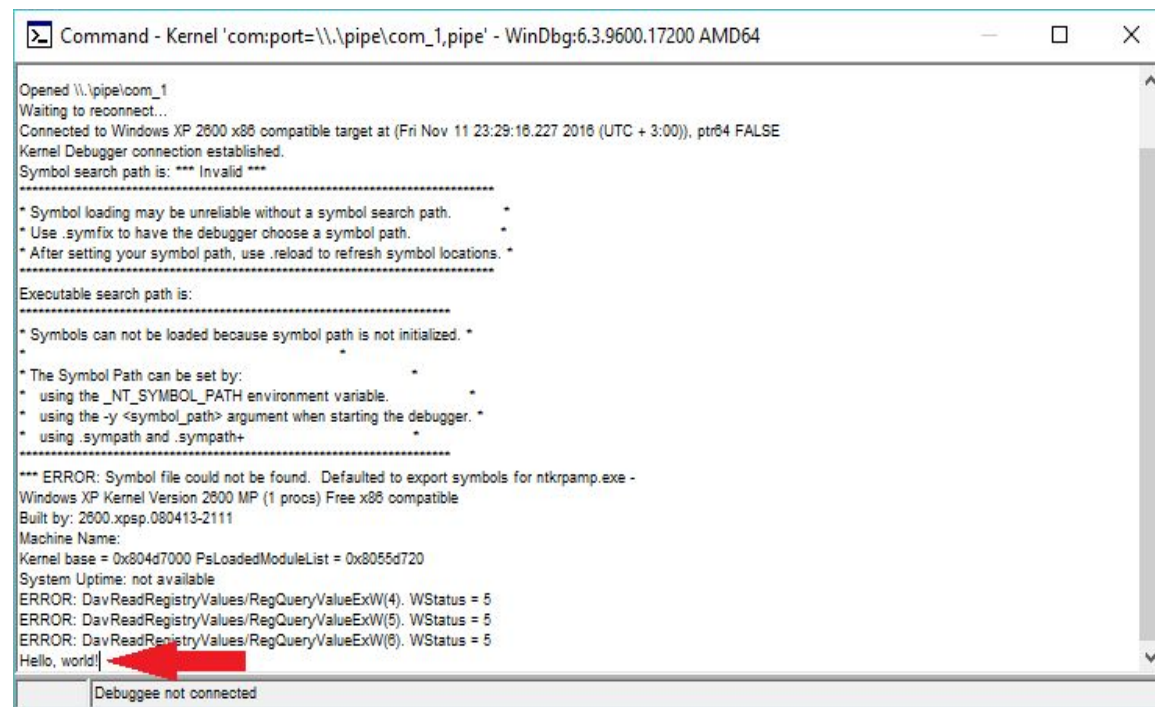
```
sc start Driver
```

Если обе операции прошли успешно, то в окне WinDbg вы увидите “Hello, world!”, а значит, драйвер написан и корректно работает. Для остановки и удаления нужно в командной строке выполнить команды:

```
sc stop Driver
```

```
sc delete Driver
```

Для удобства, можно создать bat-файл, удаляющий старую версию драйвера и запускающий новую.



```
Command - Kernel 'com:port=\\.\pipe\com_1,pipe' - WinDbg:6.3.9600.17200 AMD64

Opened \\.\pipe\com_1
Waiting to reconnect...
Connected to Windows XP 2600 x86 compatible target at (Fri Nov 11 23:29:16.227 2016 (UTC + 3:00)), ptr64 FALSE
Kernel Debugger connection established.
Symbol search path is: *** Invalid ***

-----
* Symbol loading may be unreliable without a symbol search path.
* Use .symfix to have the debugger choose a symbol path.
* After setting your symbol path, use .reload to refresh symbol locations.
-----

Executable search path is:
-----
* Symbols can not be loaded because symbol path is not initialized.
*

* The Symbol Path can be set by:
* using the _NT_SYMBOL_PATH environment variable.
* using the -y <symbol_path> argument when starting the debugger.
* using .sympath and .sympath+
-----

*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntkramp.exe -
Windows XP Kernel Version 2600 MP (1 procs) Free x86 compatible
Built by: 2600.xpsp.080413-2111
Machine Name:
Kernel base = 0x804d7000 PsLoadedModuleList = 0x8055d720
System Uptime: not available
ERROR: DavReadRegistryValues/RegQueryValueExW(4). WStatus = 5
ERROR: DavReadRegistryValues/RegQueryValueExW(5). WStatus = 5
ERROR: DavReadRegistryValues/RegQueryValueExW(6). WStatus = 5
Hello, world!
Debuggee not connected
```

# Шаг 6. Отладка драйвера

---

Для наполнения драйвера копируем в проект содержимое файла-приложения к лабораторной «Driver.c». Чтобы поставить точку останова в некотором месте программы, нужно вставить в это место строчку `DbgBreakPoint()`. Данная функция состоит из одной команды – `int 3`, известной из курса языков ассемблера.

Например, вызываем эту функцию перед первым `DbgPrint()`. Прodelываем последовательность действий из шага 5 и запускаем драйвер, не забывая удалить предыдущую версию. Когда выполнение дойдет до `DbgBreakPoint()`, система зависнет и управление передастся отладчику. Далее, аналогично отладке обычной программы (пошагово; выполнение до курсора; выполнение до следующей точки) проходим программу. Расставляя таким образом точки останова в программе, можно его отладить.

WinDBG показывает исходный код драйвера, но только при одном условии: выполняется именно та версия драйвера, которая была скомпилирована последней (то есть версия запущенного драйвера и `pdb`-файла совпадают). Поэтому при внесении каких-либо изменений в код компилируйте, копируйте и запускайте драйвер.

# Шаг 6.Отладка драйвера

---

Для удобства отладки можно настроить WinDBG под свои нужды (выберите пункт меню View , где расположены такие средства, как просмотр регистров, редактор памяти и др.) В приложении к лабораторной лежит файл windbgdark.reg. Если принять эти установки реестра (возможно, в учебной аудитории этого сделать не получится из-за ограничения в правах), то внешний вид WinDBG примет более удобный вид: темные цвета и удобное расположение окон.

# Шаг 7. Исправление ошибки в драйвере

---

Работа предлагаемого Вам драйвера заключается в перехвате системной функции NtCreateFile.

Перехватчик просматривает имя файла и, если его расширение '.xxx', то заменяет на 'yyy' и оповещает об этом консольную клиентскую программу.

Если создать текстовый файл (с любым содержимым) и сохранить его как, например, 'C:\example.xxx', то появится файл 'C:\example.yyy'

Однако, если сделать то же самое, но сохранить файл на рабочий стол в папку example, то есть по адресу 'C:\Documents and Settings\Администратор\Desktop\example\example.xxx', то имя файла никак не изменится и, соответственно, оповещения клиентской программе не поступит. Очевидна ошибка в работе драйвера, работоспособность которого зависит от внешних факторов.

В качестве одного из методов обнаружения ошибки драйвера, можно предложить пройти пошагово весь алгоритм его работы, однако в существующих драйверах кода очень много, поэтому поступим более рационально, рассуждая и выдвигая гипотезы.



# Шаг 7. Исправление ошибки в драйвере

---

*Гипотеза №1*

Предположение: Некорректно перехватилась функция NtCreateFile.

Опровержение: Если сохраняем файл в корне диска C:, то все отработывает корректно, что противоречит предположению.

# Шаг 7. Исправление ошибки в драйвере

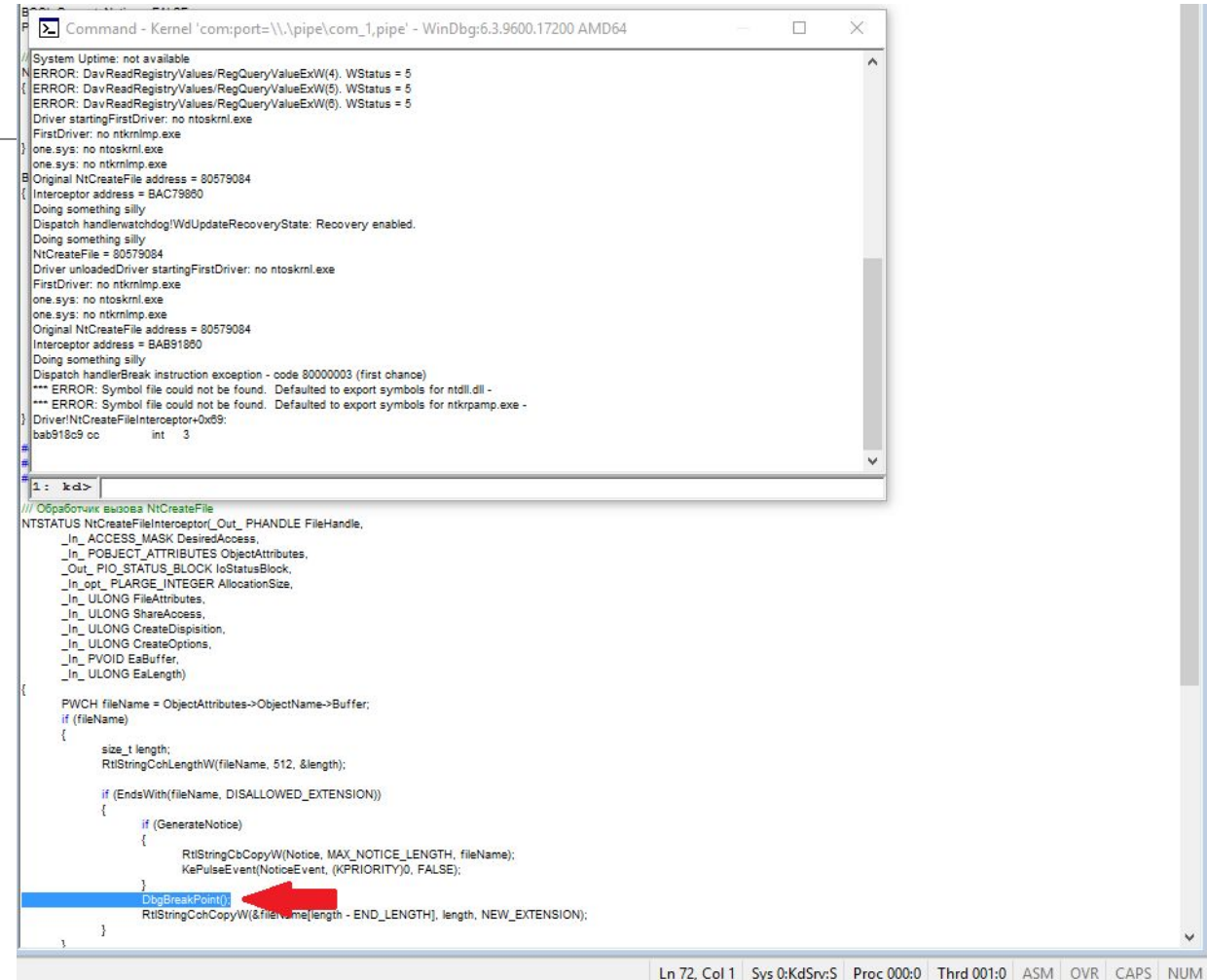
## Гипотеза №2

Предположение: Драйвер находит этот файл, но по некоторым причинам не изменяет его расширение.

Проверка гипотезы: Установим точку останова на строку 72 файла NtCreateFileInterceptor.c. В этом месте расширение заменяется на 'yyy'.

Запустим драйвер на обоих примерах (в принципе достаточно на втором, но для надёжности следует запускать и корректный вариант).

Результат: В первом случае отладчик остановился на поставленной точке как это показано на скриншоте, во втором - нет. То есть до этого места управление не дошло. Гипотеза не подтвердилась.



```
Command - Kernel 'com:port=\\.\pipe\com_1_pipe' - WinDbg:6.3.9600.17200 AMD64

// System Uptime: not available
N ERROR: DavReadRegistryValues/RegQueryValueExW(4), WStatus = 5
{ ERROR: DavReadRegistryValues/RegQueryValueExW(5), WStatus = 5
{ ERROR: DavReadRegistryValues/RegQueryValueExW(6), WStatus = 5
Driver startingFirstDriver: no ntoskrnl.exe
FirstDriver: no ntkrnlmp.exe
one.sys: no ntoskrnl.exe
}
one.sys: no ntkrnlmp.exe
Original NtCreateFile address = 80579084
{ Interceptor address = BAC79860
Doing something silly
Dispatch handler!WdUpdateRecoveryState: Recovery enabled.
Doing something silly
NtCreateFile = 80579084
Driver unloadedDriver startingFirstDriver: no ntoskrnl.exe
FirstDriver: no ntkrnlmp.exe
one.sys: no ntoskrnl.exe
one.sys: no ntkrnlmp.exe
Original NtCreateFile address = 80579084
Interceptor address = BAB91860
Doing something silly
Dispatch handler!Break instruction exception - code 80000003 (first chance)
*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntdll.dll -
*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntkrnlmp.exe -
} Driver!NtCreateFileInterceptor-0x69:
bab918c9 cc int 3

1: kd>

// Обработка вызова NtCreateFile
NTSTATUS NtCreateFileInterceptor_Out_PHANDLE FileHandle,
_In_ ACCESS_MASK DesiredAccess,
_In_ POBJECT_ATTRIBUTES ObjectAttributes,
_Out_ PIO_STATUS_BLOCK IoStatusBlock,
_In_opt_ PLARGE_INTEGER AllocationSize,
_In_ ULONG FileAttributes,
_In_ ULONG ShareAccess,
_In_ ULONG CreateDisposition,
_In_ ULONG CreateOptions,
_In_ PVOID EaBuffer,
_In_ ULONG EaLength)
{
    PWCH fileName = ObjectAttributes->ObjectName->Buffer;
    if (fileName)
    {
        size_t length;
        RtlStringCchLengthW(fileName, 512, &length);

        if (EndsWith(fileName, DISALLOWED_EXTENSION))
        {
            if (GenerateNotice)
            {
                RtlStringCbCopyW(Notice, MAX_NOTICE_LENGTH, fileName);
                KePulseEvent(NoticeEvent, (KPRIORITY)0, FALSE);
            }
            DbgBreakPoint();
            RtlStringCchCopyW(&fileName[length - END_LENGTH], length, NEW_EXTENSION);
        }
    }
}
```

# Шаг 7. Исправление ошибки в драйвере

---

*Гипотеза №3*

Предположение: Некорректно перехватилась функция NtCreateFile.

Проверка гипотезы: Из кода видно, что расширение будет заменено, если выполнилось условие в строке 65 файла «NtCreateFileInterceptor.c»:

```
if (EndsWith(fileName, DISALLOWED_EXTENSION))
```

Поставим точку останова перед данным блоком if и запустим драйвер. Для обоих значений отладчик остановится в этом месте. Для некорректно работающего значения выполним 2 шага без захода (F10) и обнаружим, что управление в данный блок if не заходит. Значит, функция EndsWith некорректно срабатывает и драйвер действительно пропускает этот файл. Гипотеза подтвердилась. Двигаемся дальше.

# Шаг 7. Исправление ошибки в драйвере

---

Проверим функцию `EndsWith` в файле «`NtCreateFileInterceptor.c`». В нашем случае, для поиска ошибки необходимо пошагово пройти всю функцию, включая цикл.

При сохранении файла на рабочий стол, на первой же итерации цикла вернется `FALSE`. Подумаем, почему так произошло и почему `str[strLen - substrLen] != 'x'`. Понятно, что `strLen` имеет некорректное значение. В то же время, пошаговая отладка случая, когда файл сохраняется на диск `C:` не выявляет никаких ошибок в функции `EndsWith()`. Необходимо понять, чем различаются два этих случая. Возможные варианты:

- Проблемы с русскими буквами в полном названии файла. Для проверки создадим нового пользователя 'user'. Файл сохранился в директории «`C:\Documents and Settings\user\Desktop\`». Гипотеза не опровергнута. Попробуем создать на рабочем столе папку `example` и сохранить в директорию «`C:\Documents and Settings\user\Desktop\example\`». Снова драйвер работает ошибочно.
- Неверно вычисляется длина строки с местом расположения файла. В первом случае длина строки маленькая, во втором - большая. Длину строки получает функция `RtlStringCchLengthW` из строки 27. Посмотрим ее описание на MSDN. Второй параметр - `size_t cchMax` – максимальная длина. И если строка на самом деле больше, то функция не вернет корректную длину строки. В нашей программе второй параметр равен 64. Изменим на большее значение (например, 1024). После этого исправления драйвер отработает правильно.

# ИТОГИ

---

В ходе лабораторной работы были изучены основные подходы к отладке драйверов, а также предназначенные для этого инструменты. Кроме того, на практике был отработан один из методов отладки драйверов.