Санкт-Петербургский Политехнический университет

Институт Компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Лабораторная работа №0**

**Дисциплина:** Проектирование ОС и компонентов

**Тема:** Анализ и профилирование прикладных приложений в ОС Windows и Linux

Выполнил студент гр. 13541/4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Шайтан

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. В. Душутина

(подпись)

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Санкт-Петербург

2017

**Выполнение работы:**

Цель работы:

В данной работе необходимо выполнить:

• Анализ загрузки приложения в ОЗУ

• Профилирование приложения

• Трассировку

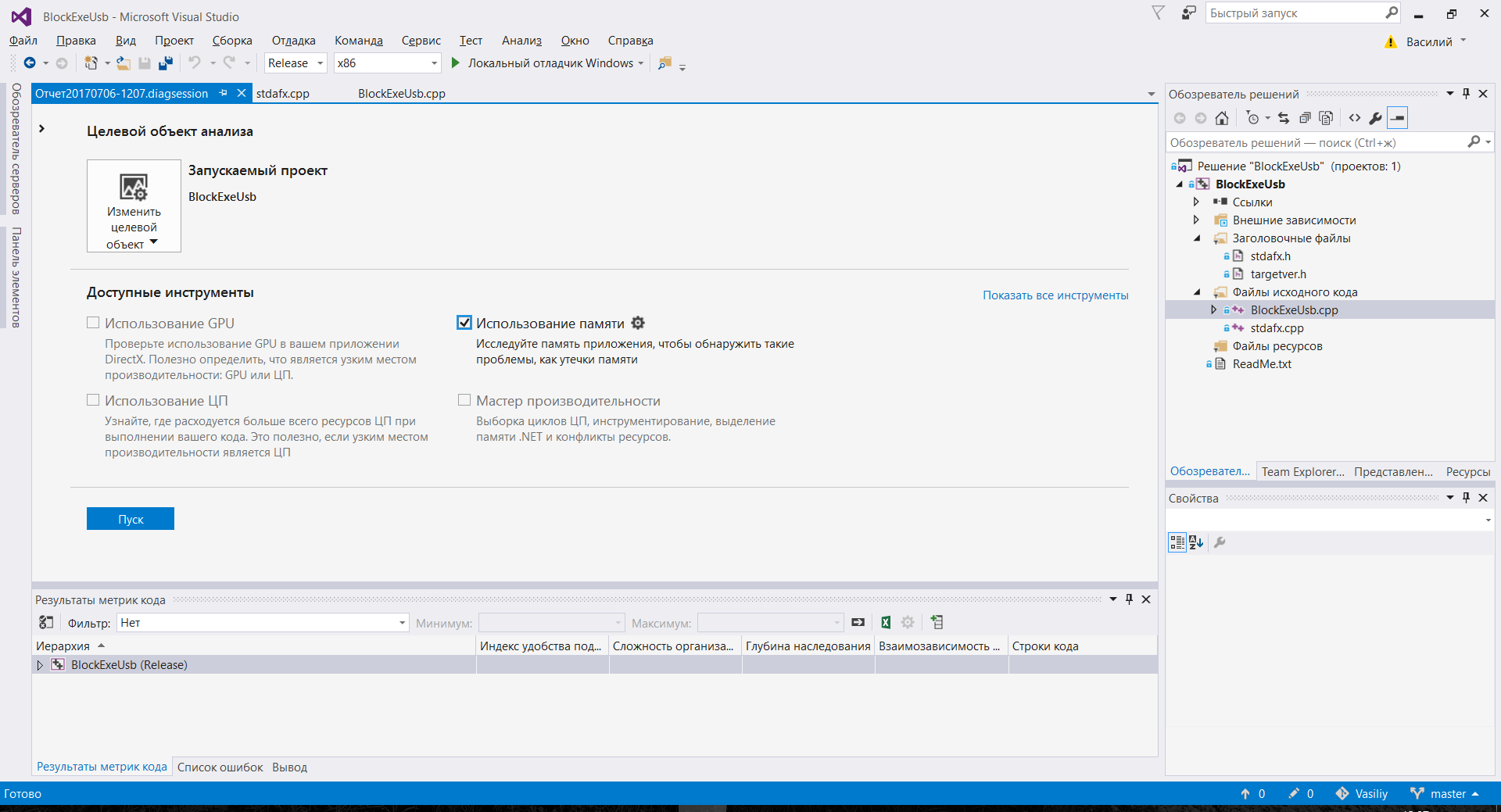
Все эти пункты необходимо выполнить по отношению к утилите, которая была написана ранее. Суть этой утилиты заключается в блокировке файлов с определенным расширением.

**Windows**

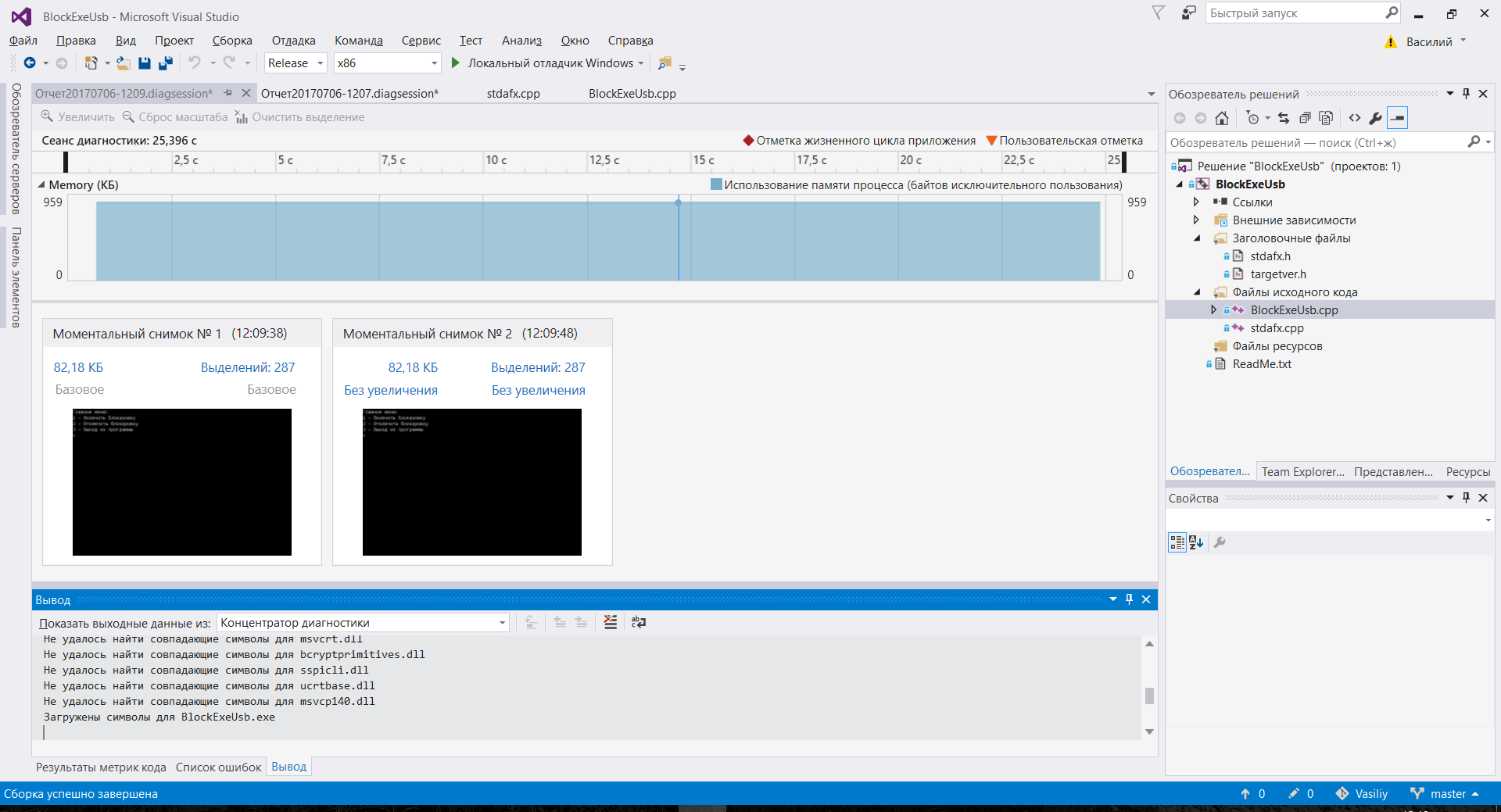
Информация о системе

|  |
| --- |
| C:\Users\Vasiliy>systeminfo  Имя узла: MSI-MSI  Название ОС: Майкрософт Windows 10 Pro (Registered Trademark)  Версия ОС: 10.0.14393 Н/Д построение 14393  Изготовитель ОС: Microsoft Corporation  Параметры ОС: Изолированная рабочая станция  Построение ОС: Multiprocessor Free  Зарегистрированный владелец: vasiliy.shaitan@gmail.com  Зарегистрированная организация:  Код продукта: 00330-80000-00000-AA051  Дата установки: 30.09.2016, 12:56:33  Время загрузки системы: 28.06.2017, 12:26:53  Изготовитель системы: Micro-Star International Co., Ltd.  Модель системы: GE70 0NC  Тип системы: x64-based PC  Процессор(ы): Число процессоров - 1.  [01]: Intel64 Family 6 Model 58 Stepping 9 GenuineIntel ~2501 МГц  Версия BIOS: American Megatrends Inc. E1756IMS.10A, 17.05.2012  Папка Windows: C:\WINDOWS  Системная папка: C:\WINDOWS\system32  Устройство загрузки: \Device\HarddiskVolume1  Язык системы: ru;Русский  Язык ввода: en-us;Английский (США)  Часовой пояс: (UTC+03:00) Москва, Санкт-Петербург, Волгоград  Полный объем физической памяти: 8 089 МБ  Доступная физическая память: 506 МБ  Виртуальная память: Макс. размер: 21 391 МБ  Виртуальная память: Доступна: 12 749 МБ  Виртуальная память: Используется: 8 642 МБ  Расположение файла подкачки: C:\pagefile.sys  Домен: WORKGROUP  Сервер входа в сеть: \\MSI-MSI  Исправление(я): Число установленных исправлений - 10.  [01]: KB3176935  [02]: KB3176936  [03]: KB3176937  [04]: KB3199209  [05]: KB3199986  [06]: KB3211320  [07]: KB4013418  [08]: KB4022730  [09]: KB4023834  [10]: KB4022715  Сетевые адаптеры: Число сетевых адаптеров - 4.  [01]: Intel(R) Centrino(R) Wireless-N 135  Имя подключения: Беспроводная сеть  Состояние: Носитель отключен  [02]: Realtek PCIe GBE Family Controller  Имя подключения: Ethernet  DHCP включен: Да  DHCP-сервер: 89.223.47.133  IP-адрес  [01]: 31.134.178.226  [02]: fe80::8081:e20e:32cc:59c4  [03]: VirtualBox Host-Only Ethernet Adapter  Имя подключения: VirtualBox Host-Only Network #2  Состояние: Оборудование не найдено  [04]: VMware Virtual Ethernet Adapter for VMnet8  Имя подключения: VMware Network Adapter VMnet8  Состояние: Оборудование не найдено  Требования Hyper-V: Расширения режима мониторинга виртуальной машины: Да  Виртуализация включена во встроенном ПО: Да  Преобразование адресов второго уровня: Да  Доступно предотвращение выполнения данных: Да  C:\Users\Vasiliy> |

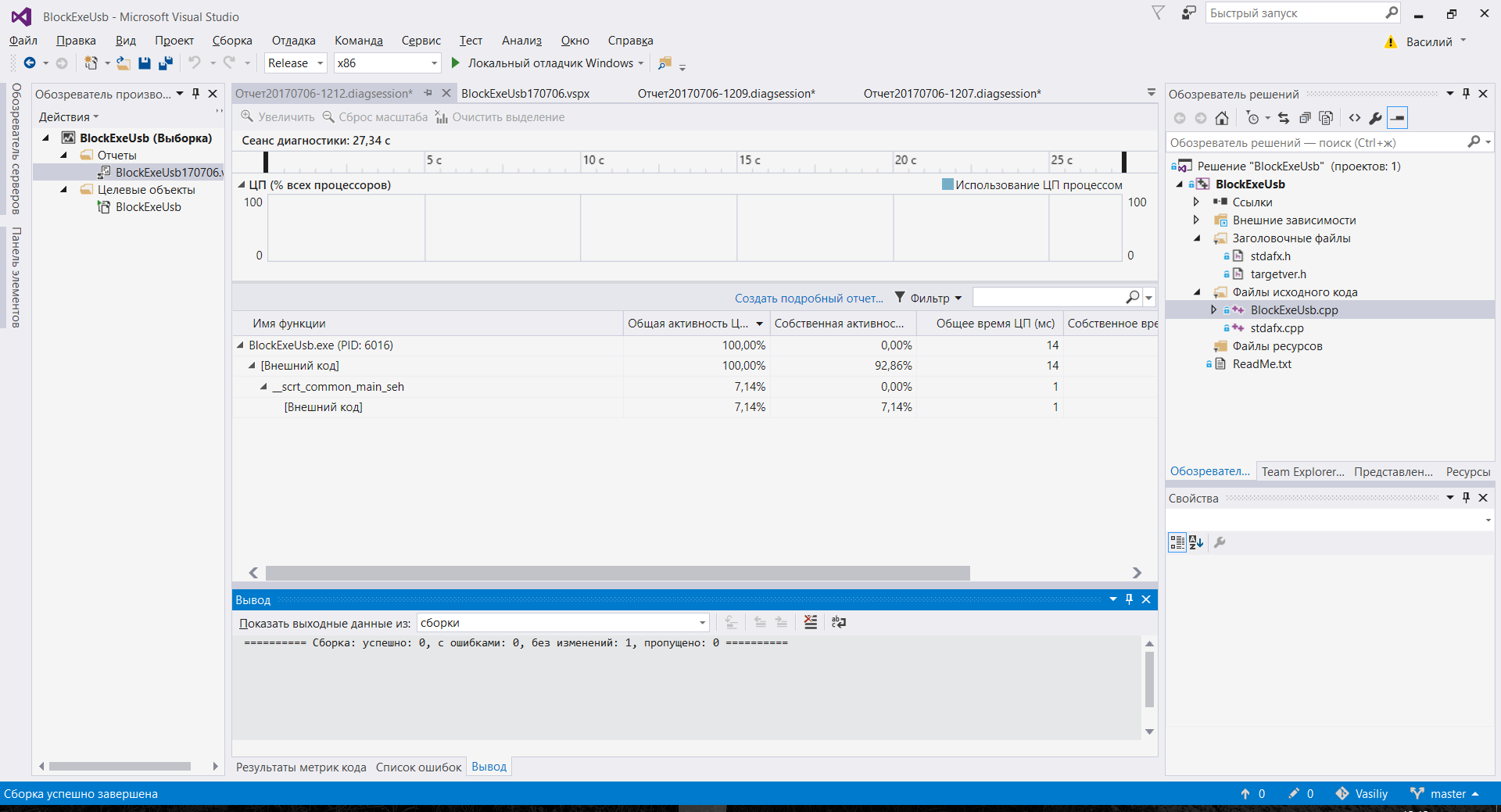
Для профилирования утилиты в системе Windows, можно использовать встроенные методы Visual Studio. Эта IDE предоставляет возможность провести несколько видов профилирования. От использования памяти до использования GPU. Чтобы запустить профилирование, во вкладке «Отладка» выбираем «Профилировщик». Но, в первую очередь, установим конфигурацию проекта в Release, т.к. в версию Debug, установленную по умолчанию, встраивается дополнительная информация для отладки приложения, и она плохо скажется на точности результатов профилирования. Далее выберем проверку использования памяти.



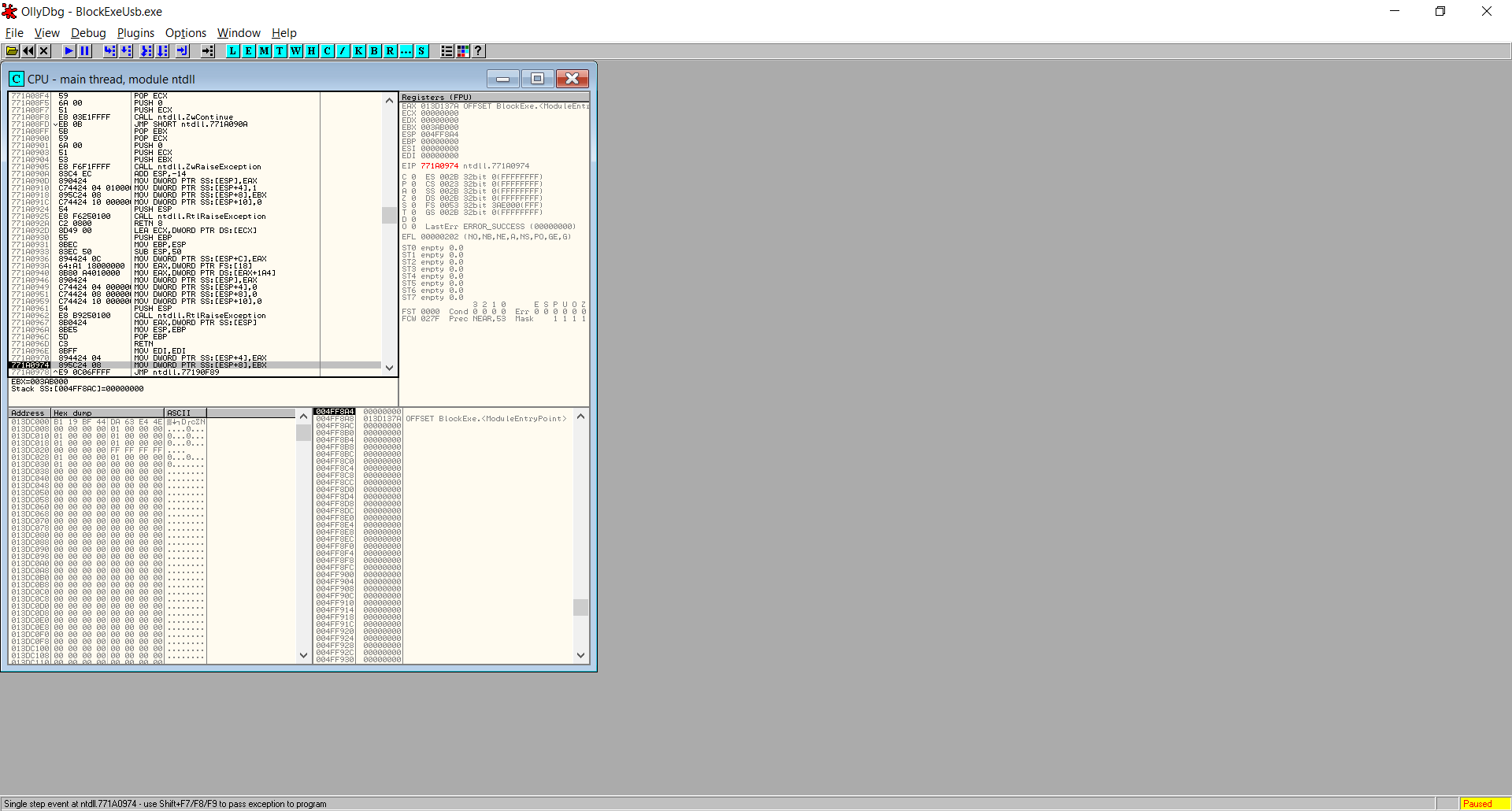
Выбираем запускаемый проект и инструмент, который хотим использовать. После запуска профилирования, мы получаем отчет расширения .diagsession c результатами проверки. Из этого файла можно увидеть, что максимальное значение используемой памяти не превышает 959 Кб. В ходе профилирования можно создавать, так называемые, «снимки», которые покажут подробную информацию об используемой памяти в ходе его создания.

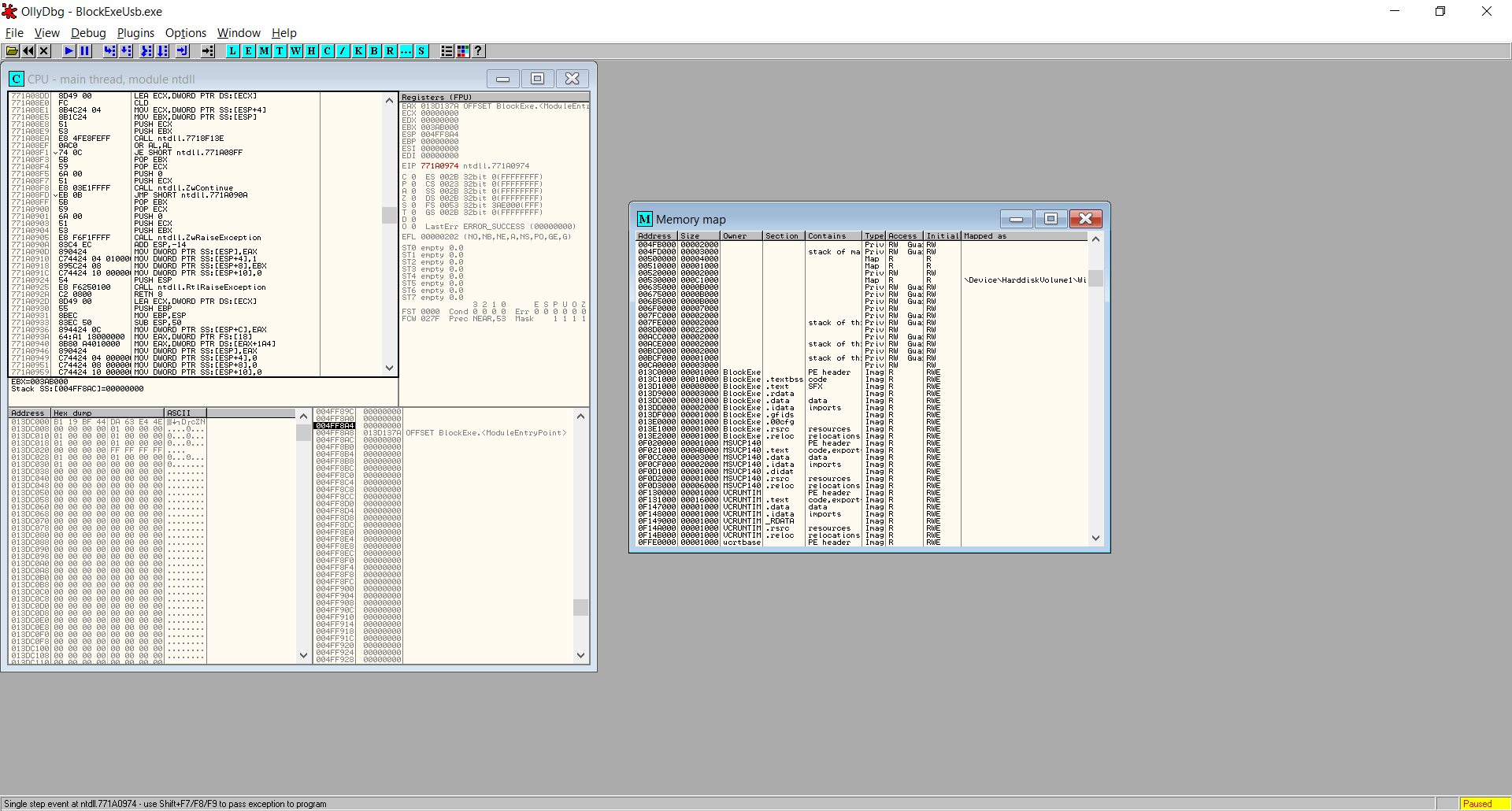


Для дальнейшего профилирования был произведен повторный запуск профилировщика с другими опциями.



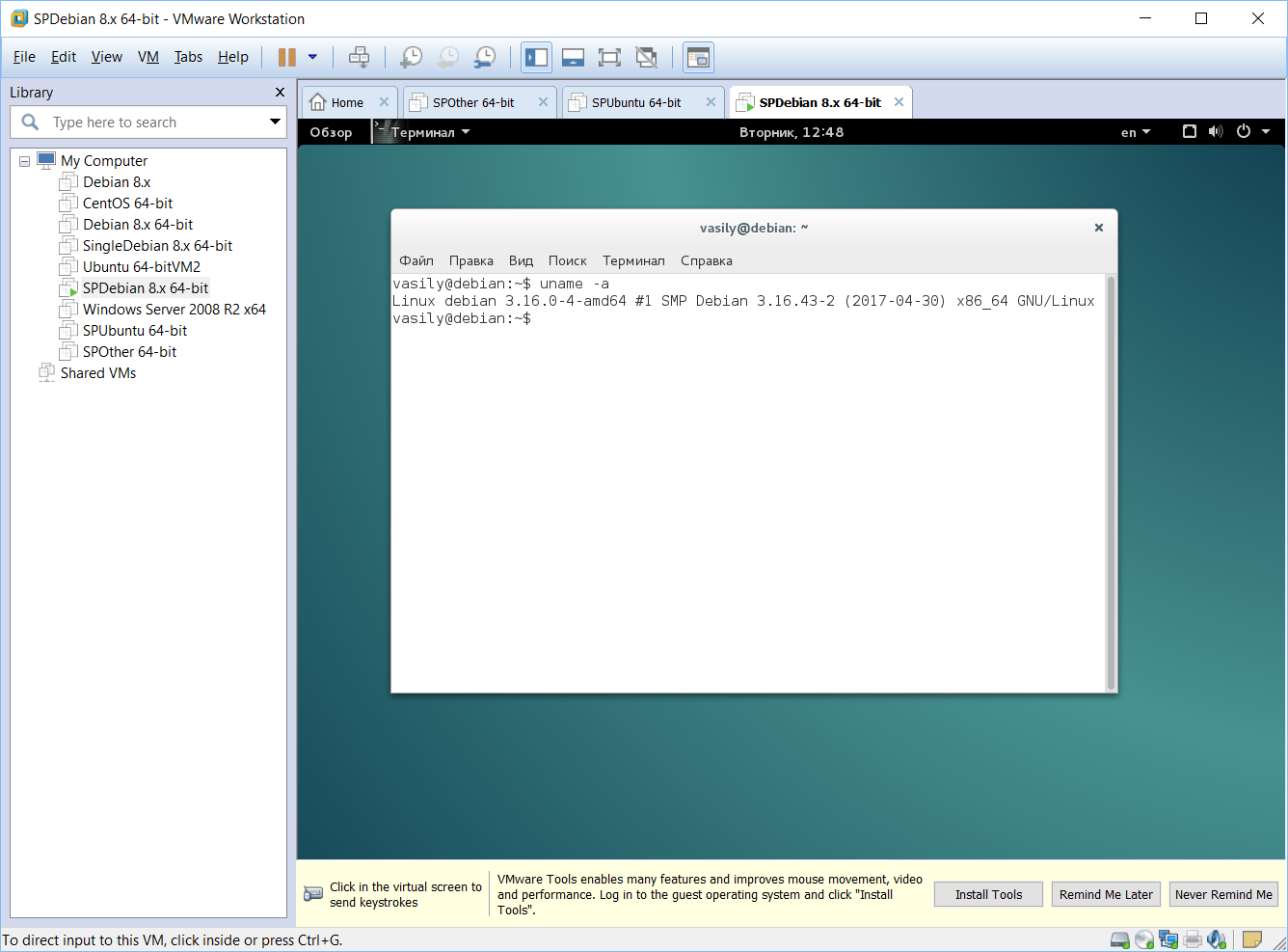
Для просмотра адреса входа программы была использована утилита OllyDBG, которая позволяет произвести дизассемблирование программы и посмотреть адрес входа.





**Linux**

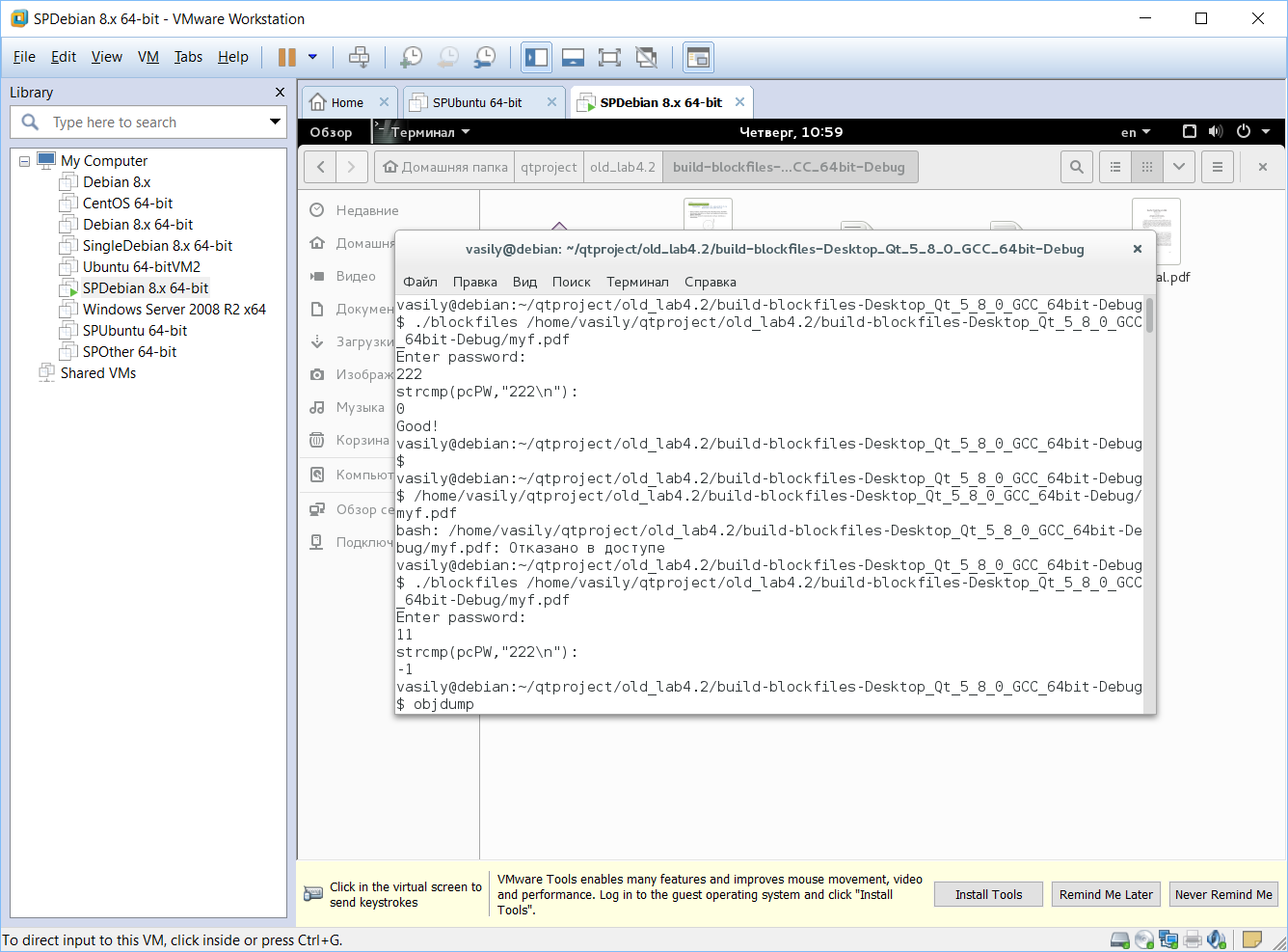
Информация о системе

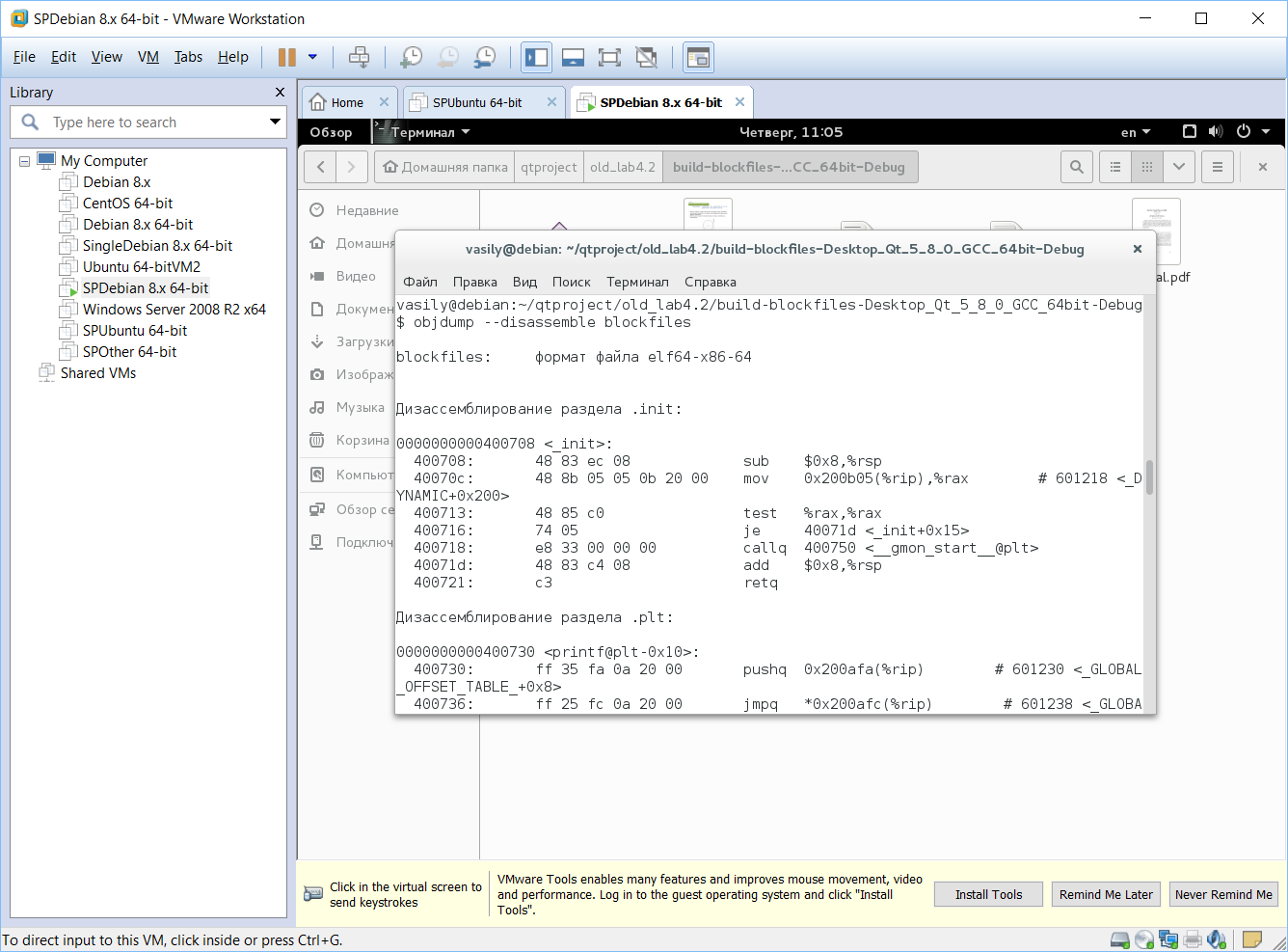


Для анализа адресного пространства процесса был использован отладчик gdb.

Помимо этого, был проанализирован дизассемблированный код утилиты с помощью objdump.

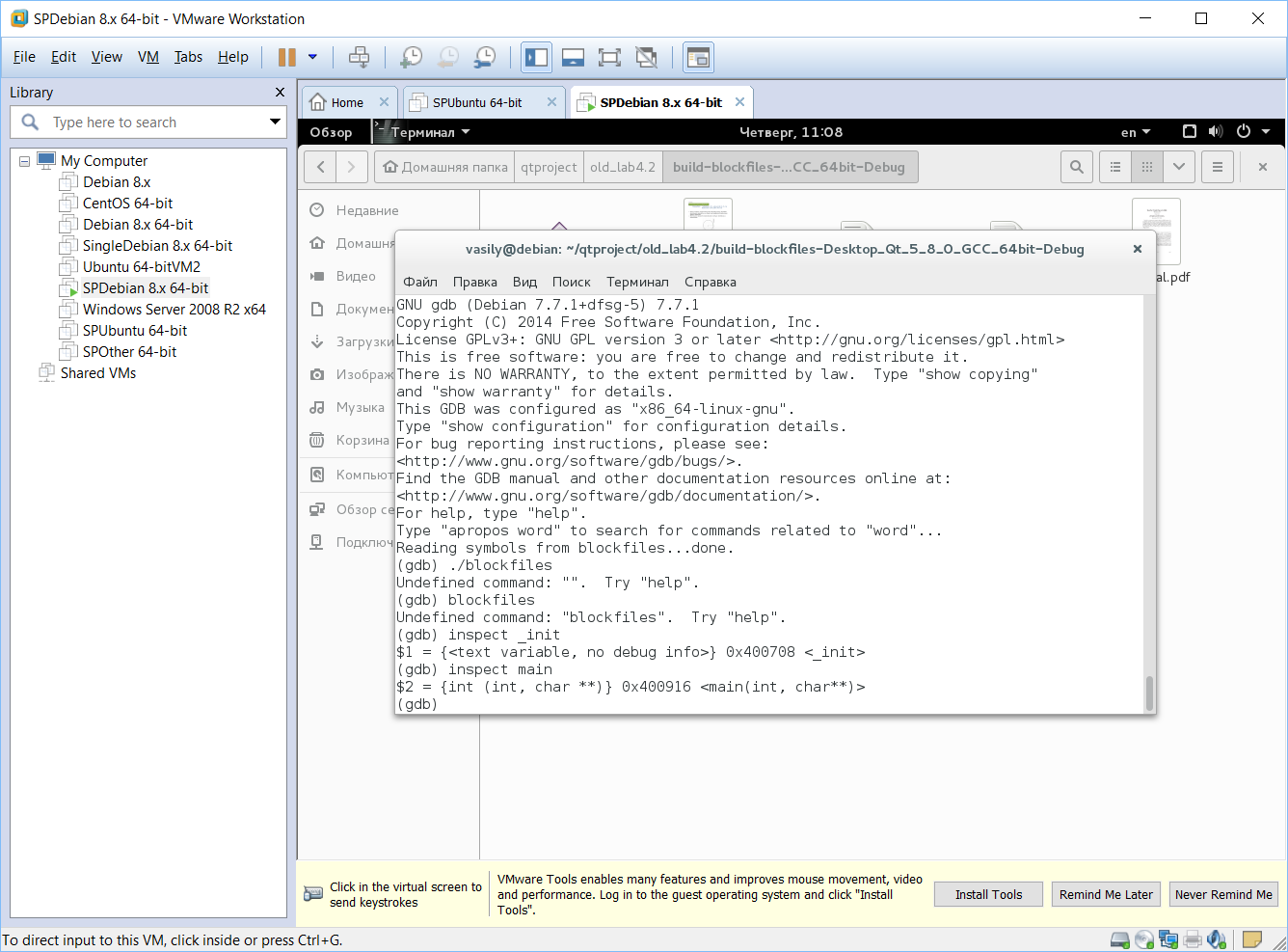
Для начала был определен адрес точки входа с помощью objdump:





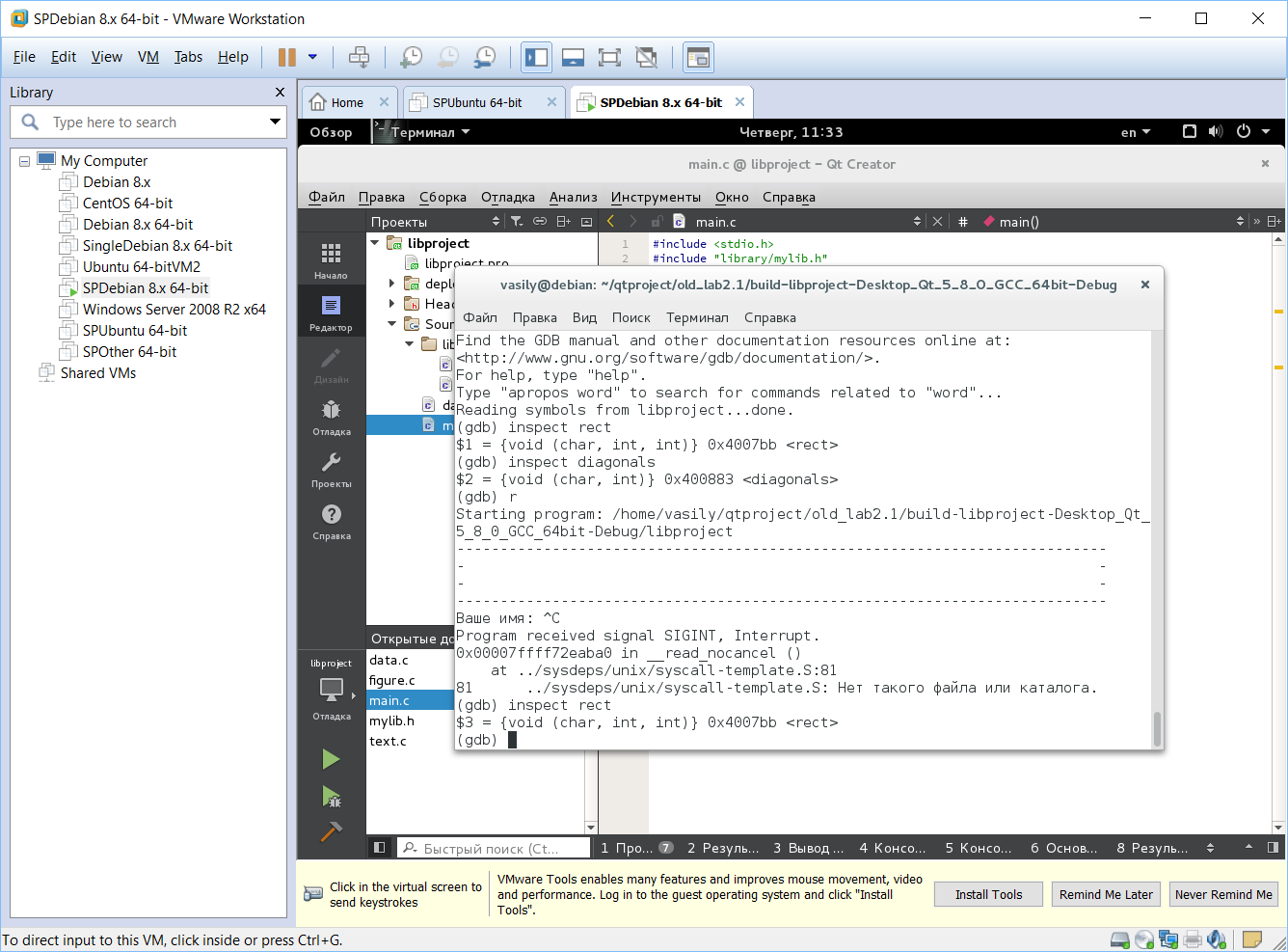
Objdump показал, что точка входа в программу располагается по адресу 400708. Для того, чтобы удостовериться в этом, был проверен адрес входа с помощью gdb:

$ gdb blockfiles

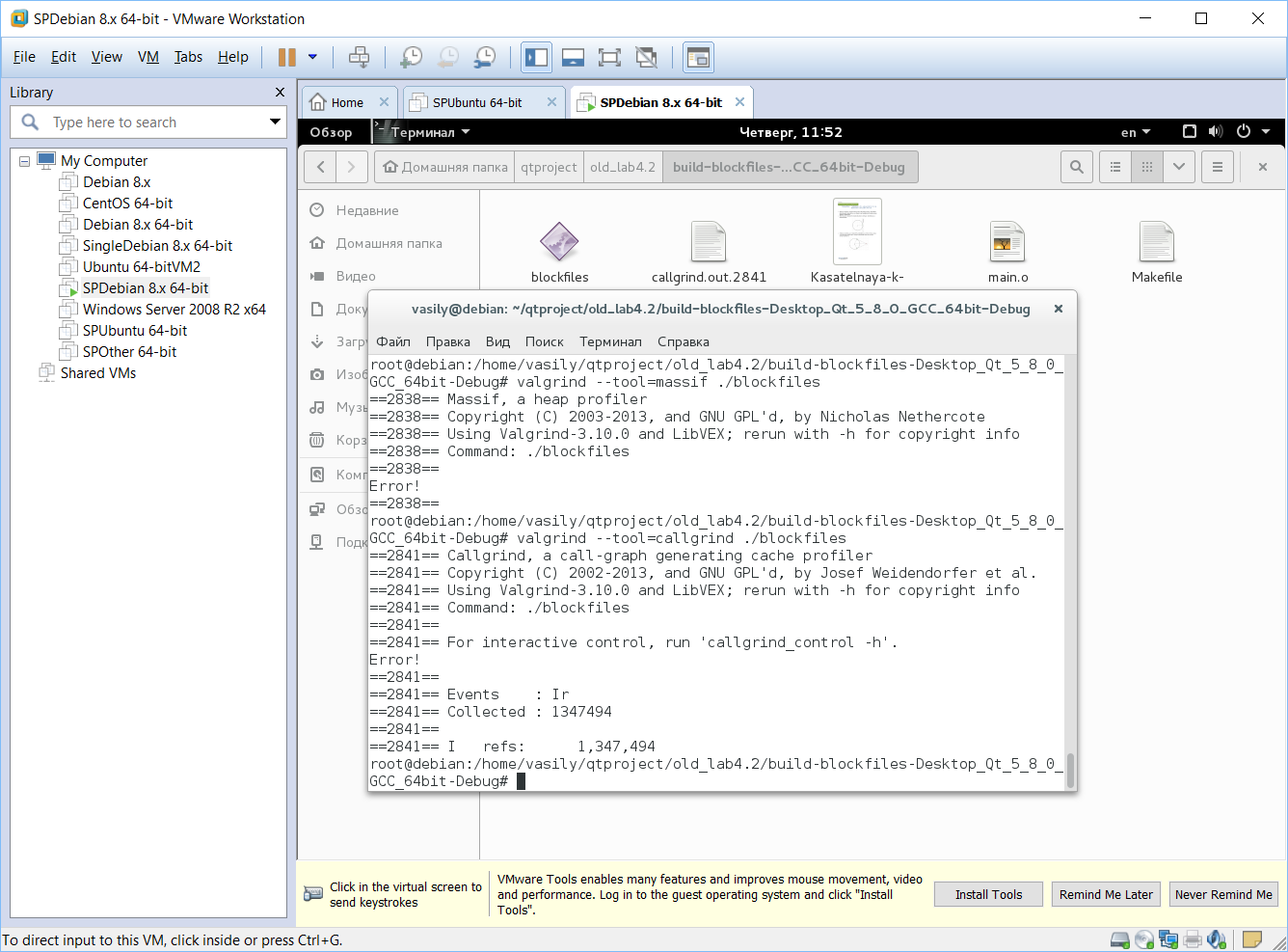


Исходя из листинга результатов выше, можно сделать вывод, что они совпадают с результатами дизассемблированного кода. С помощью gdb можно узнать адреса остальных функций, например, main.

Можно убедиться, также в том, что статическая библиотека связывается с исполняемым файлом сразу же. А динамическая — по требованию. Для этого используем inspect с именем функции из библиотеки.



Для профилирования утилиты использовалось инструментальное ПО valgrind, которое предназначено для обнаружения утечек памяти, а также профилирования. Запуск анализа производится следующим образом:



**Вывод**

В данной работе были изучены основы анализа прикладных утилит в ОС Windows и Linux. Сначала было проанализировано адресное пространство программы. С помощью инструментов отладки и дизассемблирования были определены адреса точек входа программ. Так же было проанализировано распределение статических и динамических библиотек в адресном пространстве процесса и определено, что статические библиотеки изначально располагаются в памяти процесса, а динамические подгружаются в память во время выполнения. Результаты в обеих ОС были схожи.

Кроме этого было произведено профилирование программ.

**Список используемых источников:**

1. <http://alexott.net/ru/linux/valgrind/Valgrind.html> - Что такое valgrind и зачем он нужен
2. <https://msdn.microsoft.com/ru-ru/library/ms182372.aspx> - Руководство по профилированию производительности
3. <http://www.ollydbg.de/> - analyzing debugger for Microsoft Windows