**Министерство образования и науки Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Российский государственный университет нефти и газа**

**(национальный исследовательский университет)**

**имени И. М. Губкина»**

**Кафедра Автоматизированных систем управления**

Отчет по лабораторной работе № 3

дисциплины ***Основы организации операционных систем***

**Управление памятью**

Группа: АС-23-04

Студент: Ханеский Ярослав Александрович

К.т.н., доцент Фридлянд Александр Михайлович

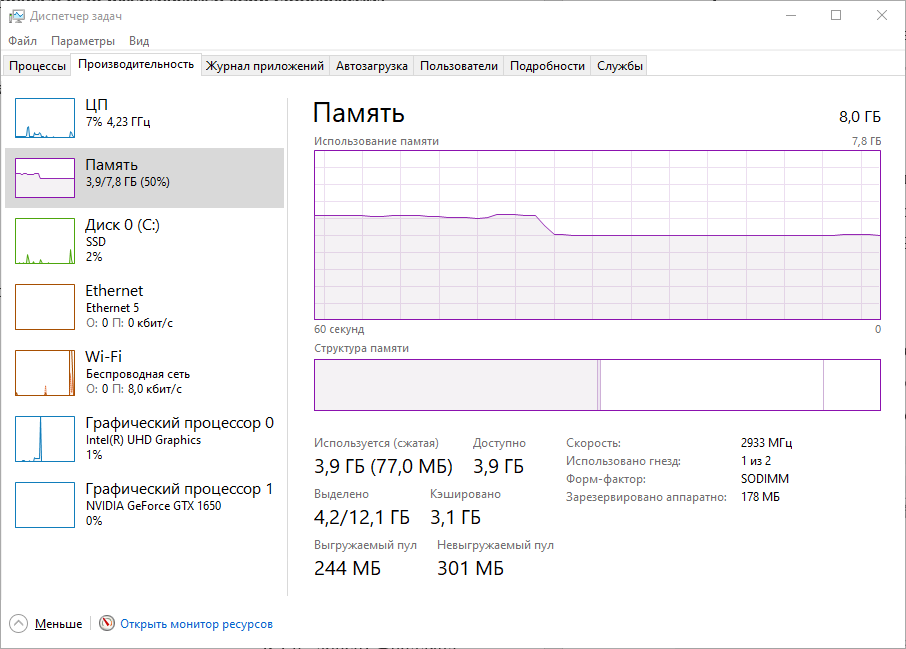
Москва

2024 г.

**Цель работы:** изучить принципы выделения памяти, анализ расхода памяти в процессе.

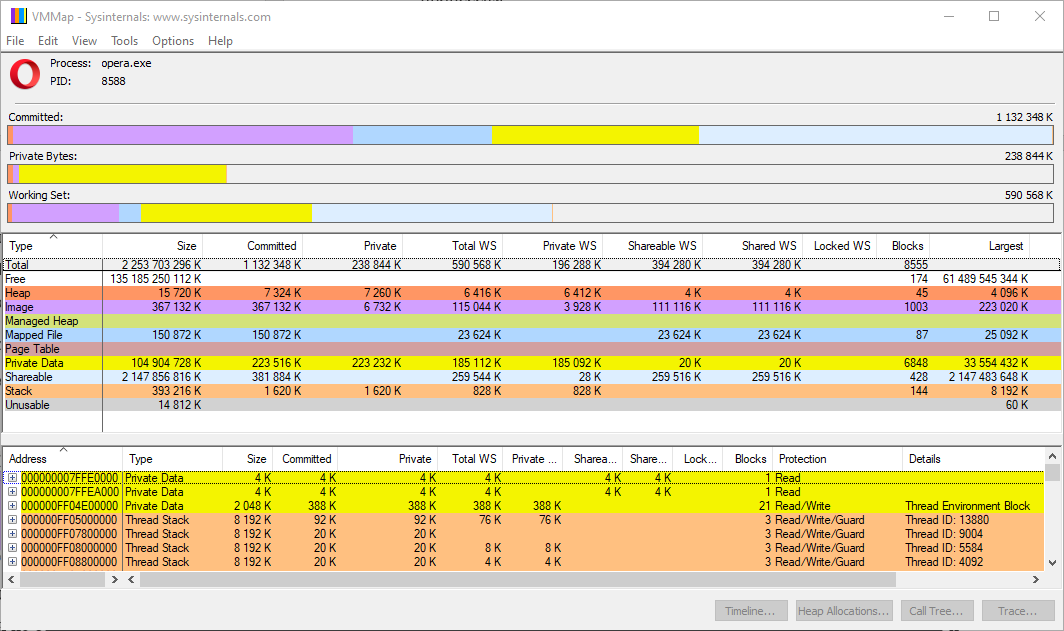
**Ход работы:**

1. С помощью встроенного в Windows системного ПО «Диспетчер задач» проведен анализ памяти основной машины:



В этом разделе отображены: объем ОЗУ в ГБ, объем физической памяти, используемый в данный момент системными ресурсами (в столбце «кэширование»), а также доступный объем памяти для использования процессами. Выгружаемый пул – это объем памяти ядра и драйвера устройства, который может перетекать из физической памяти в файл подкачки, а невыгружаемый – должен оставаться в физической памяти.

1. С помощью программы VMMap проанализируем выделение памяти процессам:



На рисунке представлена общая информация по типам выделенной памяти для процесса. К типам памяти относятся Shareable, Private Data, Image, Heap Mapped File, Stack, Page Table, также отображается объем свободной памяти.

В строке Total видно, что всего процесс зарезервировал для себя 2,1 Гб памяти, при этом закомитил из них 1106 Мб, а Working Set у него 577 Мб.

Рассмотрим каждый тип памяти по отдельности:

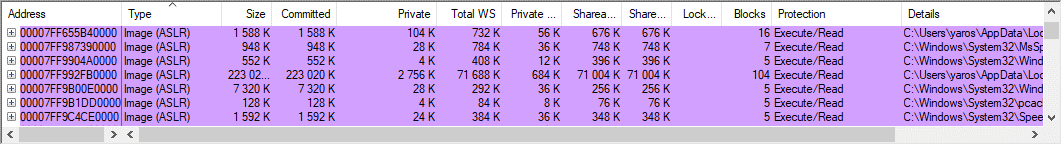
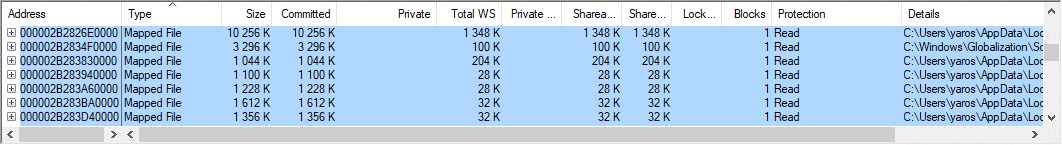
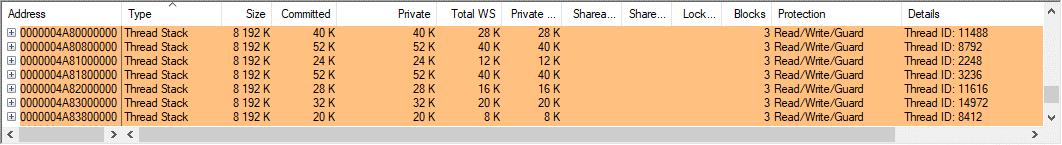


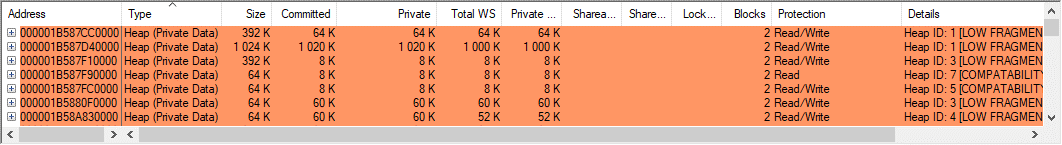
Image – память, отданная под dll и exe-файлы. Каждая dll должна быть замаплена на память приложения, так что хоть она и не будет грузиться с диска или копироваться в памяти, но она займет некоторую часть виртуальной памяти приложения. Кроме того, многие dll выделяют внутри себя память, так что они еще и расходуют Working Set приложения.



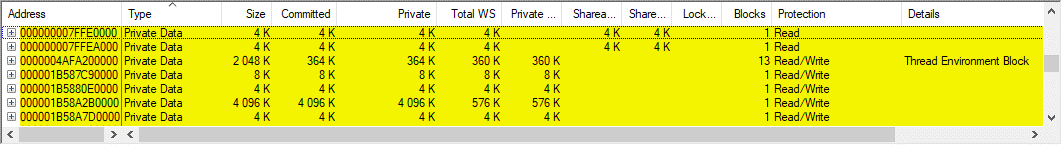
Mapped Files относится к памяти, которая связана с отображенными в память файлами (memory-mapped files). Этот механизм позволяет приложениям работать с содержимым файла так, будто оно находится в оперативной памяти, что упрощает и ускоряет доступ к данным.



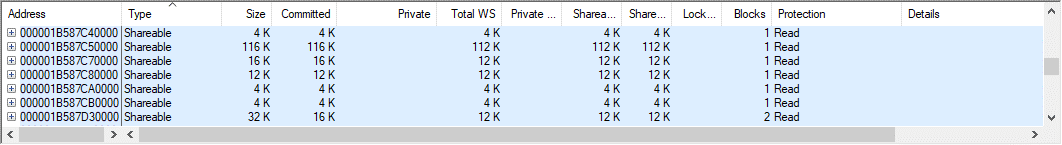
Stack – часть памяти, выделенная для каждого потока процесса, связанных с выполнением функций.



Heap – динамическая область памяти, из которой приложения могут выделять и освобождать блоки во время выполнения. Эта память используется для хранения данных, которые не могут быть заранее предсказаны или имеют переменный размер.

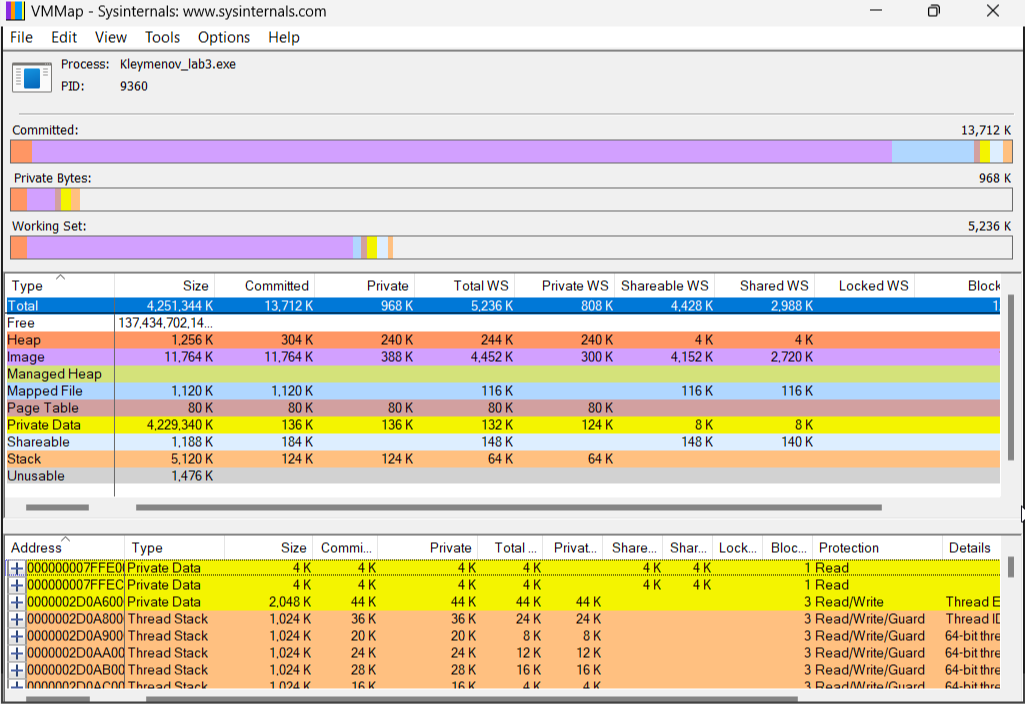


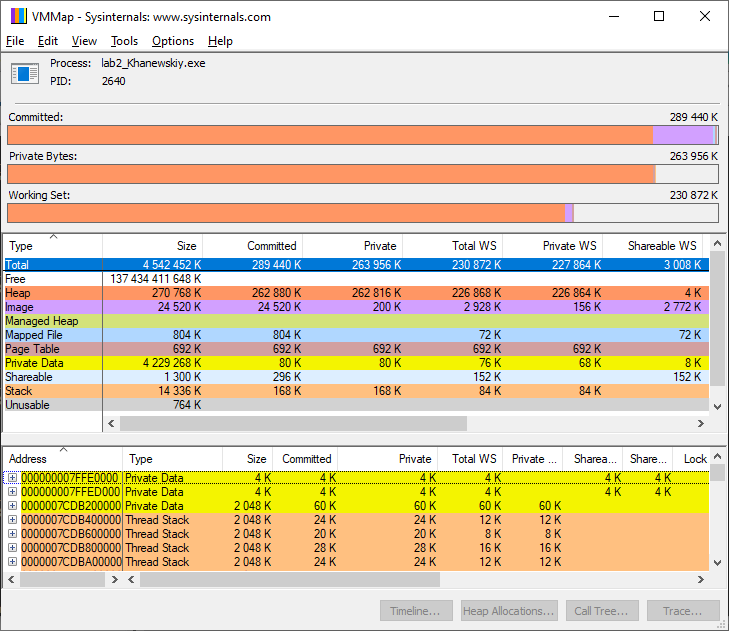
Private – это уникальная область памяти, доступная только текущему процессу. Она используется для хранения данных, которые являются специфичными для приложения и не предназначены для совместного использования.



Shareable – память, которая может быть использована несколькими процессами одновременно. Это данные, которые находятся в общем доступе и позволяют нескольким процессам читать или, реже, изменять их содержимое.

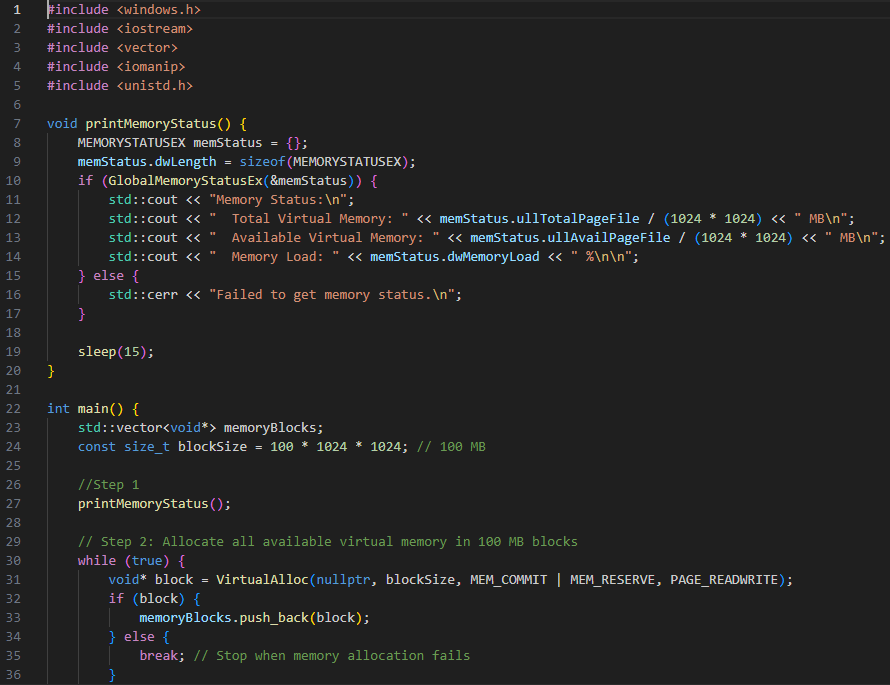
1. Запустим программу из лабораторной работы №2 в VMMap и проведем полный анализ всех ресурсов:

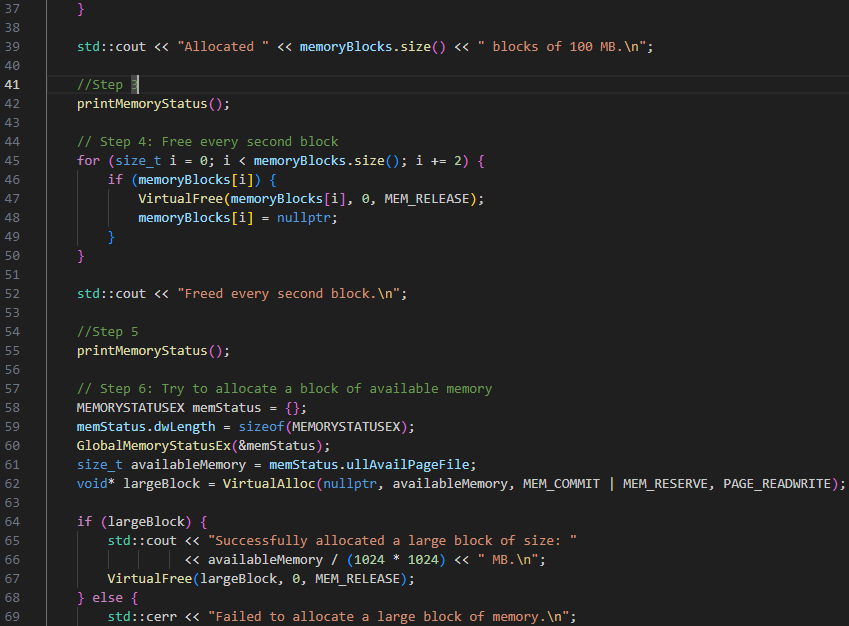


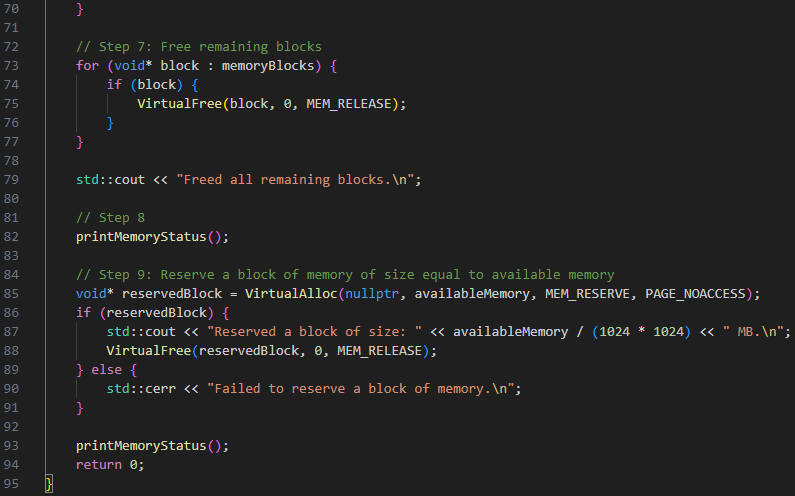


Больше всего памяти занимает тип Heap, так как во время выполнения программы в экземпляры класса stringstream помещаются полное содержимое текстовых файлов. После него идет тип Image, так как он содержит исполняемый код и статические данные программы. Далее память выделяется для каждого потока процесса, связанного с выполнением функций(Stack). Private Data уходит на стеки, кучи и другие приватные структуры данных. Оставшаяся память распределяется между Shareable, Mapped File и Page Table.

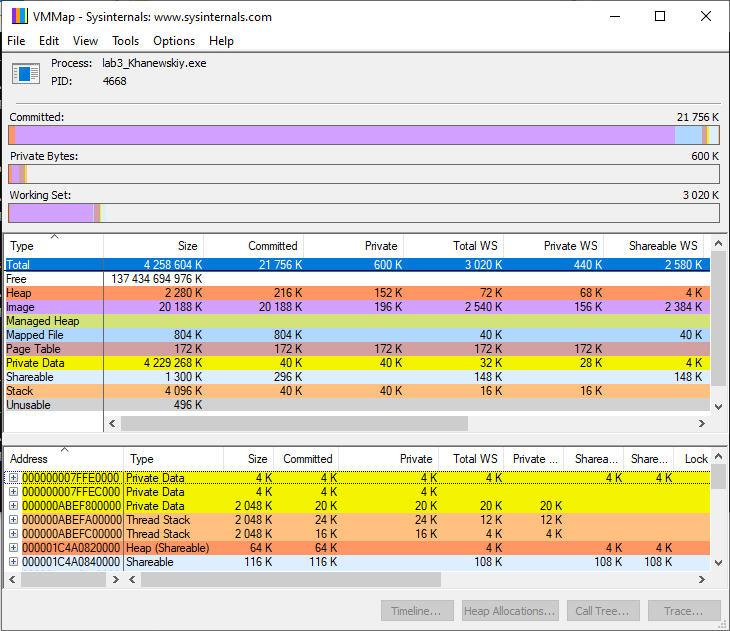
1. Реализуем программу для изучения фрагментации виртуальной памяти:



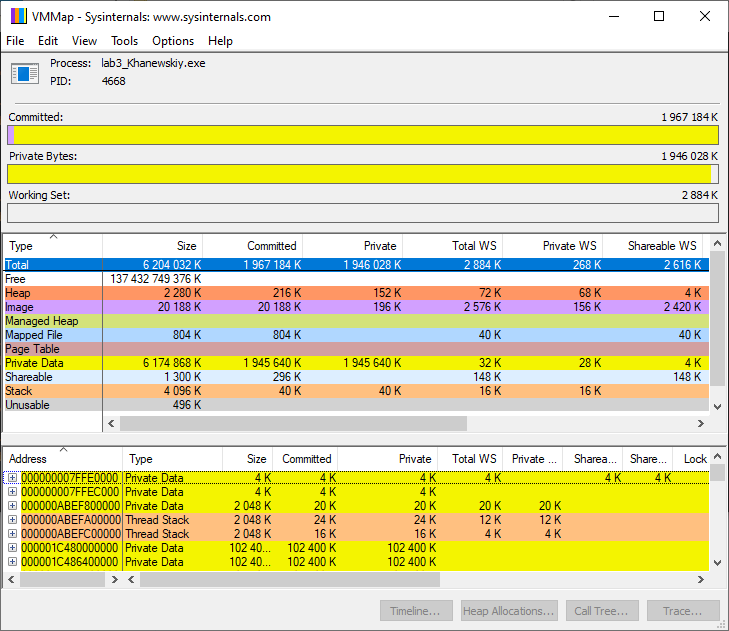




Выделение памяти на пункте 1:

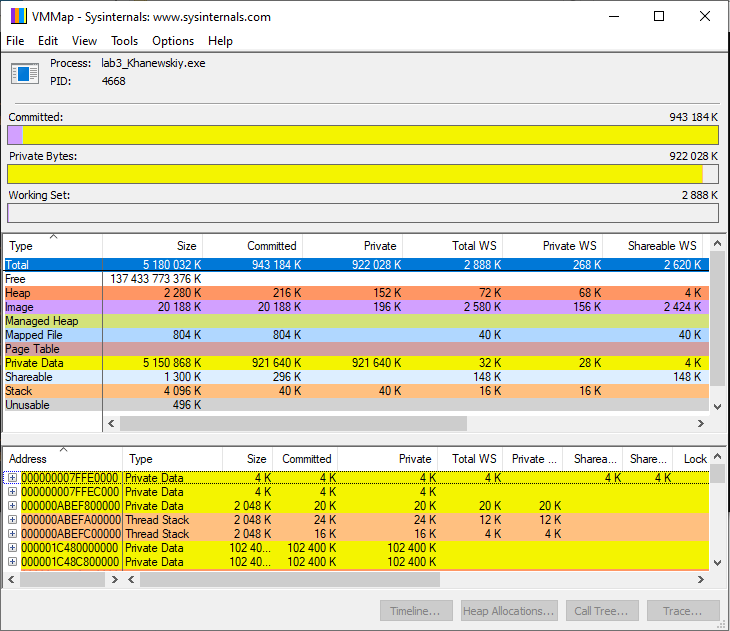


Выделение памяти на пункте 3:



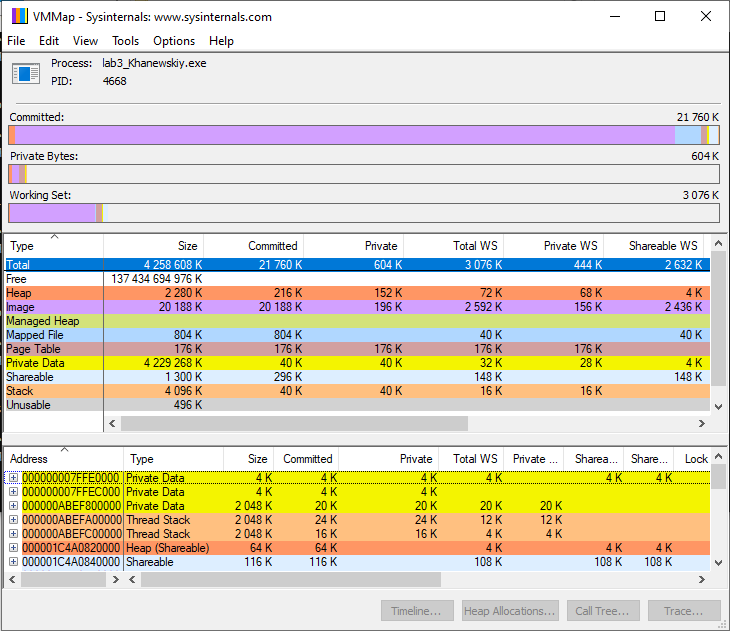
После использования всей доступной ВП, выделяемой функцией VirtualAlloc() областей размером по 100 МБ, видно, что увеличился как общий размер Private Data, так и размер закоммиченной области.

Выделение памяти на пункте 5:



Освобождение каждого второго из выделенных в пункте 2 блоков при помощи функции VirtualFree () привело к уменьшению закоммиченной области Private Data чуть больше, чем в 2 раза, а также к уменьшению на такой же объём общего размера Private Data. Также в 2 раза уменьшился размер Stack, так как в векторе memoryBlocks, хранящем указатели на функции, был удалён каждый второй элемент.

Выделение памяти на пункте 8:



Освобождение всех блоков привело к полному очищению закоммиченной области Private Data и к уменьшению на такой же объём общего размера Private Data.

Вывод программы:

