Сам объект CreateFileMapping не использует физическую память, а только резервирует виртуальное пространство.  
Основной рост потребления физической памяти происходит после MapViewOfFile и особенно после записи в память (memset).  
UnmapViewOfFile освобождает часть физической памяти, но виртуальное адресное пространство остаётся зарезервированным.

**PAGE\_READWRITE** в CreateFileMapping() - память можно как читать, так и записывать, и Windows рассматривает её как разделяемую.

**FILE\_MAP\_READ** в MapViewOfFile() -позволяет процессу только читать данные. Однако отображение всё равно идёт через разделяемую память, поскольку другие процессы тоже могут открыть этот же mapping.

Windows оптимизирует работу с MMF и рассматривает отображённый файл как общую память, потому что теоретически другой процесс тоже может открыть этот mapping. Это снижает расход Private Bytes и позволяет нескольким процессам работать с одними и теми же данными. Данные загружаются в WS Shareable / WS Shared. Когда вы вызываете MapViewOfFile(), Windows подгружает нужные страницы файла в кэш-память. Эта память считается разделяемой, потому что, если другой процесс откроет тот же mapping, он будет использовать те же страницы памяти.

Windows использует ленивую подгрузку (Lazy Loading). Данные загружаются в Working Set (WS) только тогда, когда они реально запрашиваются. При этом Windows старается не использовать Private Bytes, если можно хранить данные в общем пространстве (WS Shareable).

Как сделать память Private?

Если вы хотите, чтобы данные не попадали в WS Shareable, вам нужно сделать их чисто приватными: если другой процесс не должен получать доступ к данным, можно создать анонимное отображение (без CreateFile()), тогда оно останется только в Private Bytes.

Ваши данные оказались в Shareable Memory, потому что файл был отображён в память через CreateFileMapping, а это механизм разделяемой памяти в Windows. Если процесс создаёт MMF, то Windows считает, что другие процессы могут использовать эти данные, даже если фактически они этого не делают.

MAP\_PRIVATE не меняет исходный файл: изменения видны только в памяти.  
MAP\_SHARED сохраняет изменения в файл.  
msync() не влияет на MAP\_PRIVATE, но важен для MAP\_SHARED.

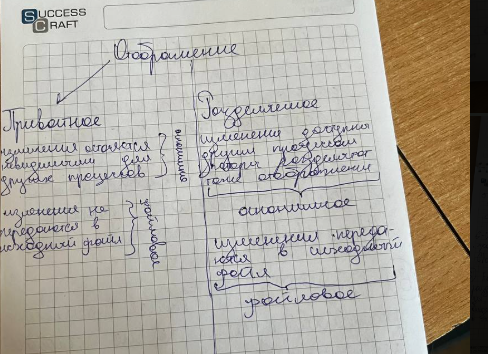
**MAP\_PRIVATE** создаёт копию содержимого файла в памяти, а не общее отображение, поэтому изменения видны только внутри процесса и не записываются в файл. > Функция msync() нужна для синхронизации изменений с файлом, но при **MAP\_PRIVATE** она не имеет эффекта для записанных данных, поскольку изменения не затрагивают сам файл.

Открывает (или создаёт) файл Lab-02d.txt.  
Отображает его в память (mmap()).  
Показывает его содержимое.  
Меняет содержимое в памяти (memcpy()).  
Пробует записать изменения (msync()), но они не сохраняются в файл.  
Освобождает ресурсы (munmap(), close()).

1. Что такое файл, отображённый в память?  
Файл, отображённый в память (memory-mapped file, MMF), — механизм, позволяющий ассоциировать содержимое файла с областью виртуальной памяти процесса. Это даёт возможность работать с файлом так же, как с массивом в памяти, без необходимости явного чтения или записи через системные вызовы.  
Механизм, который позволяет резервировать регион адресного пространства и передавать ему физ. память, физ. память берется из файла на диске (вирт. память)

2. Windows: что такое объект «проекции файла»?  
«проекции файла» (file mapping object) — объект ядра, который связывает файл с областью виртуальной памяти процесса.   
  
3. Windows: что такое представление отображения?  
Представление отображения (mapped view) — область виртуальной памяти, связанная с объектом проекции файла. Процесс может использовать представление для чтения и записи данных в файл, работая с ним как с обычным массивом.   
  
4. Windows: какие бывают отображения?  
В Windows бывают два основных типа отображений:  
Файловое отображение (File Mapping) — используется для работы с файлами без явного чтения/записи.  
Анонимное отображение (Anonymous Mapping, Section Mapping) — создаётся без привязки к файлу, используется для межпроцессного взаимодействия (IPC) и совместного использования памяти между процессами.

5. Windows: какие функции входят в API для управления файлами, отображёнными в память?  
CreateFile — создание или открытие объекта ядра файл.  
CreateFileMapping — создаёт объект проекции файла (возвращает дескриптор объекта). Сообщает какой объем физ памяти нужен для проекции файла  
MapViewOfFile — создаёт представление отображения. Возвращает указатель на представление файла. Проецирование файловых данных на адресное пространство процесса.  
UnmapViewOfFile — удаляет представление. Отключение файла данных от адресного пространства процесса.  
FlushViewOfFile — записывает на диск диапазон байтов в пределах отображенного представления файла.  
CloseHandle — закрывает объект проекции файла и файл.

6. Linux: какие бывают отображения и в чём их особенности?  
Файловые (File-backed mapping) — данные хранятся в файле, а изменения могут быть синхронизированы с диском.  
Анонимные (Anonymous mapping) — не привязаны к файлу и существуют только в оперативной памяти (например, для выделения больших массивов или разделяемой памяти).  
Главная особенность — работа с этими отображениями через mmap, позволяющая гибко управлять доступом и защитой памяти.  
  
7. Linux: какие функции входят в API для управления файлами, отображёнными в память?  
mmap — создаёт отображение в виртуальном адресном пространстве процесса. С возвращ. адрес отображения.  
munmap — удаляет отображение из виртуального адресного пространства.  
mprotect — изменяет права доступа к отображённой области.  
msync — синхронизирует разделяемое отображение с отображенным файлов. (изменения становятся видимыми)  
  
8. Что такое Copy-On-Write (COW)?  
Copy-On-Write (COW) — техника управления памятью, при которой копирование данных выполняется только при их модификации. Например, когда создаётся дочерний процесс с помощью fork(), он получает копию памяти родителя, но реальные данные копируются только в момент их изменения. Это снижает накладные расходы и улучшает производительность.

---

Результаты по проектам:  
**Lab-02a и Lab-02b (отображение файлов в память)**  
Если отображение создавалось только для чтения, то данные попадали в Shared Commit. Если отображение было для записи, то данные оказывались в Private Bytes, так как изменения были локальными для процесса до явной синхронизации с диском. При записи в файл и сбросе данных в хранилище увеличивался Working Set, так как данные загружались в оперативную память. КОГДА СОЗДАЕТСЯ ПРЕДСТАВЛЕНИЕ ТОГДА И ВЫРАСТАЕТ WORKING SET  
**Lab-02c (файл подкачки, 64 КиБ представления)**  
Данные в Shared Commit, потому что они размещаются в файле подкачки и могут быть доступны другому процессу. Если процесс закрылся и данные не были переданы другому процессу, то они остались в Private Bytes.  
**Lab-02d и Lab-02e (Linux-аналог, работа с разделяемой памятью)**В случае приватного отображения — данные в Private Bytes.  
В случае разделяемого отображения — данные в Shared Commit.  
В момент вызова msync() (сохранение в файл) Working Set мог кратковременно увеличиваться.

**CPU (Использование процессора)**  
Показывает процент загрузки процессора процессом.  
**Private Bytes (Приватные байты)**Показывает объем памяти, выделенной исключительно этому процессу и не разделяемой с другими процессами.  
Если отображаем файл приватно (Lab-02d), объем Private Bytes увеличивается.  
Если выделяем динамическую память (malloc, VirtualAlloc), тоже увеличивается.  
Освобождаем ресурсы — Private Bytes уменьшаются.  
Если растёт Private Bytes, значит, процесс запрашивает больше виртуальной памяти, но это не означает, что вся она загружена в физическую оперативную память.  
**Working Set (WS) (Набор рабочих страниц)**  
Отображает количество памяти в физической оперативной памяти (RAM), которое процесс использует в данный момент.  
В момент загрузки файла в память (Lab-02a, Lab-02b) WS увеличивается.  
Если процесс активно работает с большими объемами данных, этот показатель растёт.  
Working Set = WS Private + WS Shareable  
**PID (Process ID, Идентификатор процесса)**  
Уникальный числовой идентификатор процесса.  
Может использоваться для анализа /proc/<PID>/maps в Linux  
**WS Private (Приватный рабочий набор)**  
Часть Working Set, содержащая уникальные страницы памяти, доступные только этому процессу. Это реально загруженная в RAM часть Private Bytes.  
Если файл отображается в приватном режиме (Lab-02d), WS Private увеличивается.  
Если процесс изменяет отображенную область, а затем другой процесс пытается ее прочитать, изменения могут быть не видны (если это не Shared Commit).  
**WS Shared (Разделяемый рабочий набор)**  
Часть Working Set, содержащая разделяемые страницы памяти, доступные другим процессам.  
Если файл отображается в разделяемом режиме (Lab-02c, Lab-02e), WS Shared увеличивается.  
Если второй процесс читает данные, они уже находятся в памяти, и нет необходимости загружать их заново.  
**Shared Commit (Разделяемая коммитированная память)**  
Показывает объем памяти, который может использоваться несколькими процессами. Объём виртуальной памяти, которая была выделена процессу для совместного использования.  
Если файл отображается разделяемо (Lab-02c, Lab-02e), Shared Commit увеличивается.  
Если один процесс освобождает представление, а второй продолжает работать, объем Shared Commit не уменьшается.  
Если оба процесса закрыли представление, Shared Commit падает.

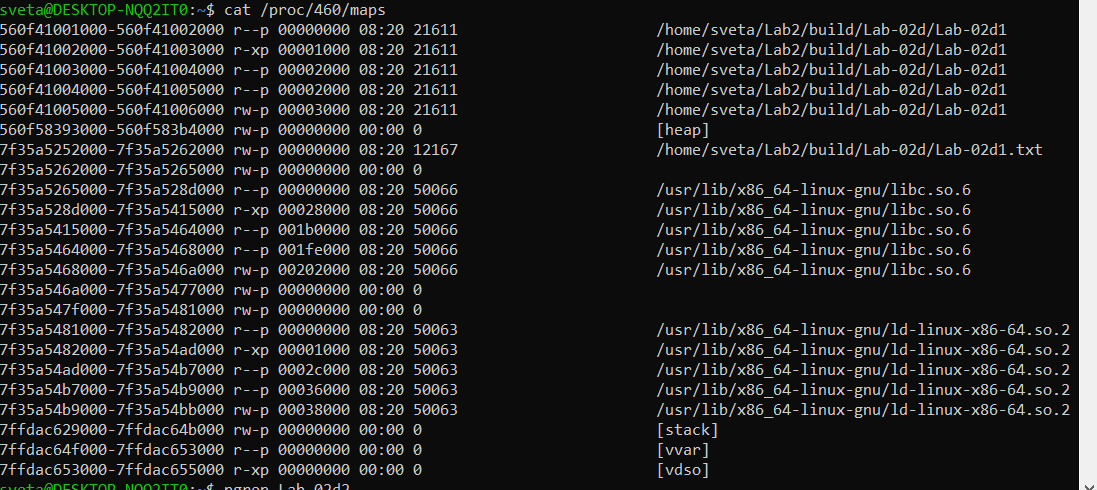
**Handles (Дескрипторы)**  
Показывает количество открытых файлов, каналов и других ресурсов процесса.  
Использование:  
Если в коде есть утечки дескрипторов, количество Handles будет расти и не уменьшаться после закрытия файлов.  
При закрытии файла CloseHandle() или munmap() Handles должны уменьшиться.  
DLLs(список загруженных библиотек)  
Столбец "Mapping" во вкладке DLLs в Process Explorer указывает на способ загрузки библиотеки в память. Он показывает, как библиотека была сопоставлена (маппирована) в адресное пространство процесса.

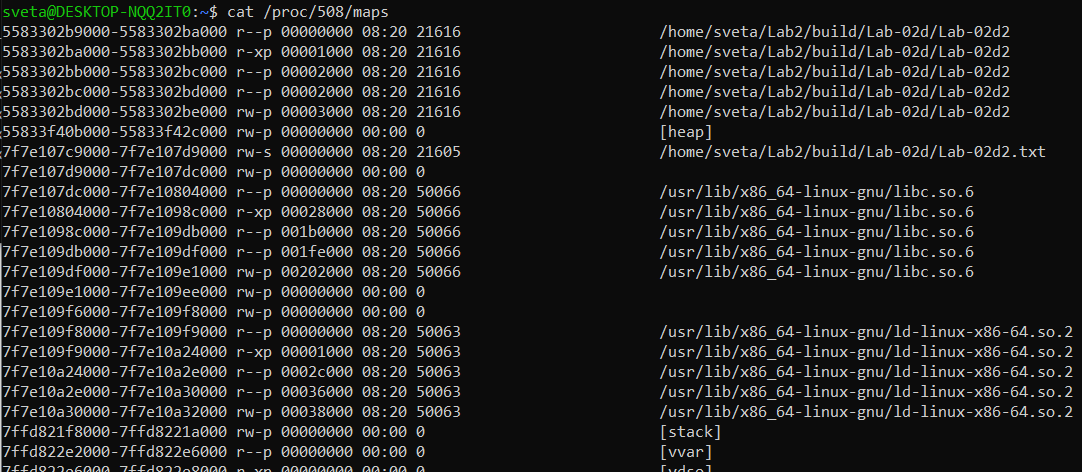
address perms offset dev inode pathname  
Указывает диапазон адресов в памяти, который занимает область.  
Права доступа к области памяти.  
Смещение в файле, с которого начинается отображение данной области.  
Указывает на устройство, с которым связана область памяти.  
Указывает на индексный узел (inode) файла, который отображается в области памяти.  
Указывает путь к файлу, если область памяти была загружена из файла.

// copy on write  
// копирования при записи  
// процесс пытается изменить, а ядро создает для него копию

pgrep Lab-02d1

cat /proc/460/maps





**Virtual Size** - сумма всех зарезервированных регионов виртуальной памяти процесса, в таких регионах располагаются все объекты - кучи, проекции файлов и разделяемая память (может подкачиваться как из файла на диске, так и из страничного файла). Показатель большой потому что включает в себя в том числе зарезервированную, но пока неиспользуемую память.

**Working Set** - это так называемый рабочий набор процесса - сумма всех страниц памяти процесса, которые находятся в ОП, а не на диске в данный момент. Размер этого набора непостоянен, если в системе достаточно много свободных физических страниц, то набор может расти вплоть до максимального и выше, если наблюдается нехватка свободных страниц, то рабочие наборы процессов начинают усекаться - у процессов отбираются странички, содержимое которых, если нужно, сбрасывается в страничный файл или файл на диске.

если, допустим, процесс отобразит в память файл на 1 Гб, и ничего с ним делать не будет, то единственным изменившимся параметром будет Virtual Size, который увеличится на этот 1 Гб  
только с той поправкой, что файл можно и читать и записывать - Virtual Size изменится

У процессов всегда присутствует **разделяемая память** (например, dll подсистемы Windows - Kernal32.dll, User32.dll и др. всегда проецируются без модификаций (по предпочтительным адресам) и поэтому разделяются всеми процессами), т.е. виртуальные адреса в разных процессах могут ссылаться на одни и те же физические страницы памяти.

механизм "копирование при записи" - пока нет модификаций физические страницы разделяются, как только происходит модификация, то процесс получает свою закрытую копию с подкачкой из страничного файла, а вот место в страничном файле в расчете на такой сценарий резервируется для каждого процесса сразу же при проецировании библиотеки.

**ТИПЫ ПАМЯТИ**

commited memory - это виртуальная память, которая физически не существует. Это зарезервированное пространство на вашем жестком диске. Эти пространства используются Windows для хранения тех задач, которые не используются процессором постоянно.

Зафиксированная память — это память, которая была привязана и закреплена в физической памяти (RAM), и операционная система не может выгрузить её в файл подкачки.

reserved memory - это часть памяти, которая отведена для определенных целей, таких как аппаратные функции или системные операции. К ней не могут получить доступ стандартные программы или процессы

Зарезервированная память — это просто память, выделенная для процесса, но в ней пока нет данных. Она не привязана к физической памяти, и в ней может не быть размещено реальных данных. Она действует как защитный механизм, гарантируя критически важным системным процессам и данным доступ к необходимой им памяти, даже когда работают другие программы.

Свободная память (free memory) - это объем памяти, который сейчас ни для чего не используется. По этой причине, особенно на серверах, удобно воспринимать свободную память, как тратящуюся впустую. После того, как ваши приложения/процессы были запущены и прошло значительное время безотказной работы, это число почти всегда должно быть небольшим.

Отображение представлений позволяет процессам экономить адресное пространство, потому что в память отображаются только представления объекта секции, необходимого в данный момент.