[https://support.microsoft.com](https://support.microsoft.com/)

**Управление виртуальной памятью в ОС Windows 10.**

**Введение**

Прямой доступ к оперативной памяти компьютера является нежелательным по следующим причинам:

* Внешняя фрагментация доступной памяти. Для борьбы с ней было бы желательно иметь возможность размещать логически непрерывные объекты (например, образы процессов или разделяемых библиотек) в несмежных областях физической памяти и/или передвигать объекты по памяти.
* Защита программ друг от друга. Из-за ошибок программирования одна задача может испортить код и данные других задач или ядра системы. Для решения этой проблемы хотелось бы гарантировать, что задача сможет работать только с той памятью, которая была выделена ей системой.
* Защита системы от злонамеренных программ. Если пользовательские задачи смогут бесконтрольно читать и изменять данные ядра системы, вызывать любые функции ядра или самостоятельно исполнять операции ввода/вывода, никакие средства защиты и разделения доступа, реализованные на уровне ядра ОС, не могут быть эффективны.
* Дисковая подкачка (свопинг). Поскольку не все задачи активны одновременно, некоторые из задач можно остановить и сбросить их области на диск и, таким образом, увеличить объем реально доступной памяти. Более того, поскольку далеко не каждой задаче в каждый момент времени нужны весь ее код и данные, очень привлекательна была бы возможность сбрасывать на диск отдельные объекты, которые задача в данный момент не использует.

Основной способ решения этой задачи в современных вычислительных системах— виртуальная память будет подробно рассмотрен в основной части реферата.

**Определения:**

Для начала введем необходимые определения:

**Виртуальная память** ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) virtual memory) — метод [управления памятью](https://ru.wikipedia.org/wiki/Управление_памятью) [компьютера](https://ru.wikipedia.org/wiki/Компьютер), позволяющий выполнять программы, требующие больше [оперативной памяти](https://ru.wikipedia.org/wiki/Оперативная_память), чем имеется в компьютере, путем автоматического перемещения частей программы между основной памятью и вторичным хранилищем (например, [жёстким диском](https://ru.wikipedia.org/wiki/Жёсткий_диск)). Для выполняющейся программы данный метод полностью прозрачен и не требует дополнительных усилий со стороны [программиста](https://ru.wikipedia.org/wiki/Программист), однако реализация этого метода требует как аппаратной поддержки, так и поддержки со стороны операционной системы.

**Управление** — перевод объекта их текущего состояния в требуемое.

**Операционная система (ОС)** — система программ, предназначенная для обеспечения определенного уровня эффективности вычислительной системы за счет:

* автоматизированного управления ее работой;
* предоставляемых пользователям определенного рода услуг.

**OC Windows** — семейство [коммерческих](https://ru.wikipedia.org/wiki/Коммерческое_программное_обеспечение) [операционных систем](https://ru.wikipedia.org/wiki/Операционная_система) корпорации [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), ориентированных на управление с помощью [графического интерфейса](https://ru.wikipedia.org/wiki/Графический_интерфейс_пользователя).

Рассмотрим подробнее управление виртуальной памятью в ОС Windows в основной части.

**Основная часть**

Если не брать во внимание программирование на машинном языке, то можно сказать, что программист обращается к памяти с помощью некоторого набора логических имен, которые чаще всего являются символьными, а не числовыми, и для которого отсутствует отношение порядка. Другими словами, в общем случае множество переменных в программе не упорядочено, хотя отдельные переменные могут иметь частичную упорядоченность (например, элементы массива). Имена переменных и входных точек программных модулей составляют пространство символьных имен. Иногда это адресное пространство называют логическим.

С другой стороны, при выполнении программы мы имеем дело с физической оперативной памятью. Физическая память представляет собой упорядоченное множество ячеек реально существующей оперативной памяти, и все они пронумерованы, то есть к каждой из них можно обратиться, указав ее порядковый номер (адрес). Количество ячеек физической памяти ограниченно и фиксировано.

Системное программное обеспечение должно связать каждое указанное пользователем символьное имя с физической ячейкой памяти, то есть осуществить отображение пространства имен на физическую память компьютера. В общем случае это отображение осуществляется в два этапа: сначала системой программирования, а затем операционной системой. Между этими этапами обращения к памяти имеют форму виртуального адреса. При этом можно сказать, что множество всех допустимых значений виртуального адреса для некоторой программы определяет ее виртуальное адресное пространство, или виртуальную память. Виртуальное адресное пространство программы зависит, прежде всего, от архитектуры процессора и от системы программирования и практически не зависит от объема реальной физической памяти компьютера. Можно еще сказать, что адреса команд и переменных в машинной программе, подготовленной к выполнению системой программирования, как раз и являются виртуальными адресами.

Система программирования осуществляет трансляцию и компоновку программы, используя библиотечные программные модули. В результате работы системы программирования полученные виртуальные адреса могут иметь как двоичную форму, так и символьно-двоичную. Это означает, что некоторые программные модули и их переменные получают какие-то числовые значения, а те модули, адреса для которых пока не могут быть определены, имеют по-прежнему символьную форму, и их окончательная привязка к физическим ячейкам будет осуществлена на этапе загрузки программы в память перед ее непосредственным выполнением.

Если рассматривать общую схему двухэтапного отображения адресов, то с позиции соотношения объемов упомянутых адресных пространств можно отметить наличие следующих трех ситуаций:

* объем виртуального адресного пространства программы Vv меньше объема физической памяти Vp (Vv < Vp);
* ооъем виртуального адресного пространства программы Vv равен объему физической памяти Vp (Vv = Vp );
* объем виртуального адресного пространства программы Vv больше объема физической памяти Vp (Vv > Vp ).

Две первые ситуации (Vv < Vp) и (Vv = Vp ) ныне практически не встречается. В наше время мы уже достигли того, что ситуация превышения объема виртуального адресного пространства программы над объемом физической памяти (Vv > Vp) характерна для персональных компьютеров. Теперь это самая обычная ситуация, и для нее имеется несколько методов распределения памяти, отличающихся как сложностью, так и эффективностью.

Простое непрерывное распределение — это самая простая схема, согласно которой вся память условно может быть разделена на три области:

* область, занимаемая операционной системой;
* область, в которой размещается исполняемая задача;
* незанятая ничем (свободная) область памяти.

По этой схеме построена память в MS-DOS.

Для организации мультипрограммного и/или мультизадачного режима необходимо обеспечить одновременное расположение в оперативной памяти нескольких задач. Память задаче может выделяться одним сплошным участком (неразрывное распределение памяти) или несколькими порциями, которые могут быть размещены в разных областях памяти (разрывное распределение).

Начнем с методов неразрывного распределения памяти. Самая простая схема распределения памяти между несколькими задачами предполагает, что память, не занятая ядром операционной системы, может быть разбита на несколько непрерывных частей — разделов (partitions, regions). Разделы характеризуются именем, типом, границами (как правило, указываются начало раздела и его длина).

Разбиение памяти на несколько непрерывных (неразрывных) разделов может быть фиксированным (статическим) либо динамическим (то есть процесс выделения нового раздела памяти происходит непосредственно при появлении новой задачи).

Методы распределения памяти, при которых задаче уже может не предоставляться сплошная (непрерывная) область памяти, называют разрывными. Идея выделять память задаче не одной сплошной областью, а фрагментами позволяет уменьшить фрагментацию памяти, однако этот подход требует для своей реализации больше ресурсов, он намного сложнее. Если задать адрес начала текущего фрагмента программы и величину смещения относительно этого начального адреса, то можно указать необходимую нам переменную или команду. Таким образом, виртуальный адрес можно представить состоящим из двух полей. Первое поле будет указывать на ту часть программы, к которой обращается процессор, для определения местоположения этой части в памяти, а второе поле виртуального адреса позволит найти нужную нам ячейку относительно найденного адреса. Программист может либо самостоятельно разбивать программу на фрагменты, либо можно автоматизировать эту задачу, возложив ее на систему программирования.

Первым среди разрывных методов распределения памяти был сегментный. Для этого метода программу необходимо разбивать на части и уже каждой такой части выделять физическую память. Естественным способом разбиения программы на части является разбиение ее на логические элементы — так называемые сегменты.

Второй метод распределения памяти — страничный. При страничном способе организации виртуальной памяти все фрагменты программы, на которые она разбивается (за исключением последней ее части), получаются одинаковыми. Одинаковыми полагаются и единицы памяти, которые предоставляются для размещения фрагментов программы. Эти одинаковые части называют страницами и говорят, что оперативная память разбивается на физические страницы, а программа — на виртуальные страницы. Часть виртуальных страниц задачи размещается в оперативной памяти, а часть — во внешней. Обычно место во внешней памяти, называют файлом подкачки, или страничным файлом (paging file). Иногда этот файл называют swap-файлом, тем самым подчеркивая, что записи этого файла — страницы — замещают друг друга в оперативной памяти. В некоторых операционных системах выгруженные страницы располагаются не в файле, а в специальном разделе дискового пространства.

Разбиение всей оперативной памяти на страницы одинаковой величины, причем кратной степени двойки, приводит к тому, что вместо одномерного адресного пространства памяти можно говорить о двухмерном. Первая координата адресного пространства — это номер страницы, вторая координата — номер ячейки внутри выбранной страницы (его называют индексом). Таким образом, физический адрес определяется парой (Рp, i), а виртуальный адрес — парой (Pv, i), где Pv — номер виртуальной страницы, Рр — номер физической страницы, i — индекс ячейки внутри страницы. Количество битов, отводимое под индекс, определяет размер страницы, а количество битов, отводимое под номер виртуальной страницы, — объем потенциально доступной для программы виртуальной памяти. Отображение, осуществляемое системой во время исполнения, сводится к отображению Pv в Рр и приписыванию к полученному значению битов адреса, задаваемых величиной i. При этом нет необходимости ограничивать число виртуальных страниц числом физических, то есть не поместившиеся страницы можно размещать во внешней памяти, которая в данном случае служит расширением оперативной.

Для отображения виртуального адресного пространства задачи на физическую память для каждой задачи необходимо иметь таблицу страниц для трансляции адресных пространств. Для описания каждой страницы диспетчер памяти операционной системы заводит соответствующий дескриптор, который отличается от дескриптора сегмента прежде всего тем, что в нем нет поля длины — ведь все страницы имеют одинаковый размер. По номеру виртуальной страницы в таблице дескрипторов страниц текущей задачи находится соответствующий элемент (дескриптор). Если бит присутствия имеет единичное значение, значит данная страница размещена в оперативной, а не во внешней памяти, и мы в дескрипторе имеем номер физической страницы, отведенной под данную виртуальную. Если же бит присутствия равен нулю, то в дескрипторе мы будем иметь адрес виртуальной страницы, расположенной во внешней памяти. Таким образом и осуществляется трансляция виртуального адресного пространства на физическую память.

Защита страничной памяти основана на контроле уровня доступа к каждой странице. Как правило, возможны следующие уровни доступа:

* только чтение;
* чтение и запись;
* только выполнение.

Каждая страница снабжается соответствующим кодом уровня доступа. При трансформации логического адреса в физический сравнивается значение кода разрешенного уровня доступа с фактически требуемым. При их несовпадении работа программы прерывается.

При обращении к виртуальной странице, не оказавшейся в данный момент в оперативной памяти, возникает прерывание, и управление передается диспетчеру памяти, который должен найти свободное место. Обычно предоставляется первая же свободная страница. Если свободной физической страницы нет, то диспетчер памяти по одной из дисциплин замещения (LRU, LFU, FIFO, случайный доступ) определит страницу, подлежащую расформированию или сохранению во внешней памяти. На ее месте он разместит новую виртуальную страницу, к которой было обращение из задачи, но которой не оказалось в оперативной памяти.

Если объем физической памяти небольшой и даже часто требуемые страницы не удается разместить в оперативной памяти, возникает так называемая «пробуксовка». Другими словами, пробуксовка — это ситуация, при которой загрузка нужной страницы вызывает перемещение во внешнюю память той страницы, с которой мы же активно работаем. Очевидно, что это очень плохое явление. Чтобы его не допускать, желательно увеличить объем оперативной памяти (сейчас это просто, поскольку стоимость модуля оперативной памяти многократно снизилась), уменьшить количество параллельно выполняемых задач или прибегнуть к более эффективным дисциплинам замещения.

Для абсолютного большинства современных операционных систем характерна дисциплина замещения страниц LRU как самая эффективная. Однако в операционных системах Windows NT/2000/XP разработчики, желая сделать их максимально независимыми от аппаратных возможностей процессора, отказались от этой дисциплины и применили правило FIFO. А для того чтобы хоть как-то компенсировать неэффективность правила FIFO, была введена «буферизация» тех страниц, которые должны быть записаны в файл подкачки на диск или просто расформированы. Принцип буферизации прост. Прежде чем замещаемая страница действительно окажется во внешней памяти или просто расформированной, она помечается как кандидат на выгрузку. Если в следующий раз произойдет обращение к странице, находящейся в таком «буфере», то страница никуда не выгружается и уходит в конец списка FIFO. В противном случае страница действительно выгружается, а на ее место в «буфер» попадает следующий «кандидат». Величина такого «буфера» не может быть большой, поэтому эффективность страничной реализации памяти в Windows NT/2000/XP намного ниже, чем в других операционных системах, и явление пробуксовки начинается даже при существенно большем объеме оперативной памяти.

Как и в случае с сегментным способом организации виртуальной памяти, страничный механизм приводит к тому, что без специальных аппаратных средств он существенно замедляет работу вычислительной системы. Поэтому обычно используется кэширование страничных дескрипторов. Наиболее эффективным механизмом кэширования является ассоциативный кэш.

Итак, основным достоинством страничного способа распределения памяти является минимальная фрагментация. Поскольку на каждую задачу может приходиться по одной незаполненной странице, очевидно, что память можно использовать достаточно эффективно; этот метод организации виртуальной памяти был бы одним из самых лучших, если бы не два следующих обстоятельства.

Первое — это то, что страничная трансляция виртуальной памяти требует существенных накладных расходов. В самом деле, таблицы страниц нужно тоже размещать в памяти. Кроме того, эти таблицы нужно обрабатывать; именно с ними работает диспетчер памяти.

Второй существенный недостаток страничной адресации заключается в том, что программы разбиваются на страницы случайно, без учета логических взаимосвязей, имеющихся в коде. Это приводит к тому, что межстраничные переходы, как правило, осуществляются чаще, нежели межсегментные, и к тому, что становится трудно организовать разделение программных модулей между выполняющимися процессами.

**ОС Windows**

Виртуальная память в ОС Windows применяется всегда, даже если объем памяти, требуемый всем запущенным процессам, не превышает объем ОЗУ, установленный в системе.

Всем процессам (например, исполняемым файлам приложений), работающим в 32-разрядных версиях Windows, назначаются виртуальные адреса (виртуальное адресное пространство) в диапазоне от 0 до 4 294 967 295 (2\*32–1 = 4 ГБ) независимо от того, сколько фактической оперативной памяти доступно на компьютере.  
  
В конфигурации Windows по умолчанию 2 гигабайта (ГБ) этого виртуального адресного пространства выделены каждому процессу для частного использования, а другие 2 ГБ совместно применяются всеми процессами и операционной системой. Обычно приложения (например, Блокнот, Word, Excel и Acrobat Reader) используют только часть из 2 ГБ частного адресного пространства. Операционная система назначает рамки страниц ОЗУ только используемым страницам виртуальной памяти.

Оперативная память — это ограниченный ресурс, а виртуальная память для большинства практических целей не ограничена. Может существовать множество процессов, каждому из которых выделено собственное частное виртуальное адресное пространство объемом 2 ГБ. Если объем памяти, используемый всеми текущими процессами, превышает объем ОЗУ, операционная система перемещает страницы (частями по 4 КБ) одного или нескольких виртуальных адресных пространств на жесткий диск компьютера. Это освобождает ОЗУ для других целей. В системах Windows такие “выгруженные” страницы хранятся в одном или нескольких файлах Pagefile.sys в корневом каталоге раздела. В каждом разделе может быть один такой файл. Размещение и размер файла подкачки задаются в свойствах системы.

Системный монитор — это основное средство, с помощью которого можно отследить производительность системы и определить узкие места. Чтобы запустить системный монитор, нажмите кнопку Пуск, выберите пункт Панель управления, Администрирование, а затем дважды щелкните Системный монитор. Далее представлена сводка по некоторым важным счетчикам и предоставляемым им сведениям:

* Память, байт выделенной виртуальной памяти. Этот счетчик представляет меру спроса на виртуальную память.
* Процесс, рабочий набор, \_всего. Этот счетчик представляет меру активно используемой виртуальной памяти.
* Файл подкачки, используется %pagefile. Этот счетчик представляет меру фактического использования файла подкачки.
* Память, обмен страниц/с.
* Память, вывод страниц/с. Этот счетчик показывает, сколько страниц виртуальной памяти записывались в файл подкачки для освобождения страниц ОЗУ для других целей каждую секунду.
* Память, байт кэш-памяти,  
  Память, байт в невыгружаемом пуле,  
  Память, байт в выгружаемом пуле,  
  Память, всего байт системного кода,  
  Память, всего байт системных драйверов.  
  Сумма этих счетчиков представляет меру того, какая часть из 2 ГБ общей части виртуального адресного пространства объемом 4 ГБ фактически используется.
* Память, доступно МБ. Этот счетчик представляет меру того, сколько оперативной памяти доступно для удовлетворения спроса на виртуальную память.