# ОПЕРАЦИОННЫЕ СИСТЕМЫ

https//vikchas.ru

# Лекция 3 «Подсистема управления основной памятью»

Часовских Виктор Петрович
доктор технических наук, профессор
кафедры ШИиКМ, ФГБОУ ВО
«Уральский государственный
экономический университет

Екатеринбург 2026

Организация и управление основной, или первичной, оперативной, памятью (ОП) вычислительной машины один из самых важных факторов, определяющих построение ОС.

Для выполнения программ или обращения к данным необходимо, чтобы они предварительно были размещены в ОП. Вторичная, или внешняя, память— это, как правило, накопители на магнитных дисках, накопители на CD- (DVD-) дисках, накопители на магнитных лентах и т. д. В отличие от внешней памяти, ОП для сохранения информации требуется электропитание.

## Иерархия запоминающих устройств

Всю имеющуюся в компьютере память можно рассматривать как единую иерархическую

систему запоминающих устройств (ЗУ), которые отличаются такими характеристиками, как размер, среднее время доступа к данным и стоимость единицы хранения.

Выделяют 4 основных (укрупненных) уровня иерархии:

1) внутреннюю память процессора (регистры, организованные в регистровый файл, и кэш процессора);

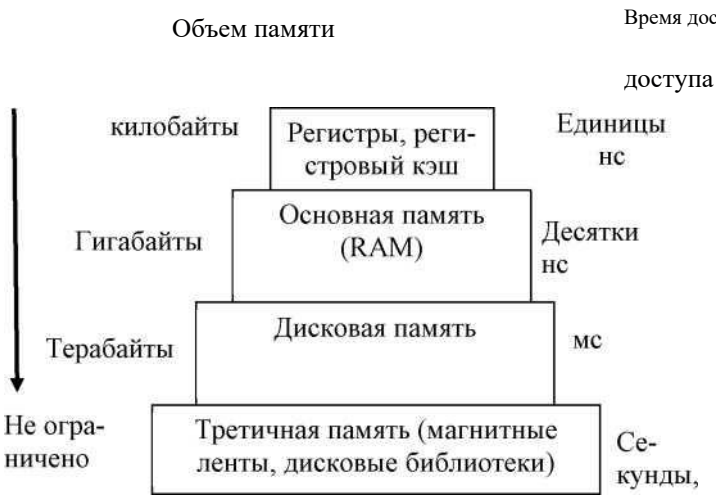
2) ОП системы и вспомогательные карты памяти;

3) накопители с «горячим» доступом, или вторичную компьютерную память — это жесткие диски и твердотельные накопители, не требующие длительных подготовительных действий для получения данных;

4) накопители, требующие переключения

носителей (Off-line bulk storage), или третичную, память. Сюда относятся магнитные ленты, ленточные и дисковые библиотеки, требующие длительной перемотки либо механического (или ручного) переключения носителей информации.

В соответствии с рис. 1 можно сделать следующие выводы: чем больше объем устройства, тем менее быстродействующим оно является; стоимость же хранения данных в расчете на один бит увеличивается с ростом быстродействия устройств.




1. Иерархия устройств памяти (обобщенная схема)

## Функции ОС по управлению основной памятью

В начале развития ЭВМ, ОП представляла собой самый дорогостоящий ресурс. В связи с этим она требовала особого внимания со стороны разработчиков систем — необходимо было обеспечить как можно более эффективное использование столь дорогостоящего ресурса. В первых ЭВМ главной задачей считалось оптимальное использование основной памяти благодаря рациональной организации и

управлению ею.

Под **организацией памяти** обычно понимают то, каким образом представляется и используется ОП: сколько программ размещается в памяти и, если их несколько, способ их размещения — одинаковое количество ячеек выделять каждой программе или только столько, сколько запрашивает; как с течением времени перераспределять память и т. д.

Функциями ОС по управлению ОП в мультипрограммной системе являются:

1) отслеживание свободной и занятой памяти;
2) выделение памяти процессам и освобождение памяти по завершении процессов;
3) вытеснение кодов и данных из ОП на диск (полное или частичное), когда физически размеры основной памяти недостаточны для размещения в ней всех процессов, и возвращение

их в ОП, когда в ней освобождается место;

4) настройка адресов процесса на конкретную область физической памяти;

5) защита областей памяти, выделенных одному процессу от доступа другого процесса (эта функция реализуется ОС в т. ч. аппаратными средствами).

## Стратегии управления памятью

Стратегии управления памятью должны быть направлены на наиболее эффективное использование ее ресурсов. Они делятся на следующие категории:

– стратегии выборки (загрузки);

– стратегии выборки по запросу (по требованию);

– стратегии упреждающей выборки;

– стратегии размещения;

– стратегии замещения.

Стратегии выборки должны определять, когда следует разместить очередной блок программы и (или) данных в ОП. Выборка по запросу предполагает, что очередной блок кода или данных загружается в ОП, когда его запрашивает выполняющийся процесс. Упреждающая выборка предполагает возможность прогнозирования, какие данные могут потребоваться в ближайшее время, и загрузку их заранее.

Стратегии размещения должны определить, куда (место ОП) следует помещать поступающий блок кода и (или) данных. Обычно это стратегии первого подходящего, наиболее подходящего или наименее подходящего по размеру свободного участка памяти.

Стратегии замещения ставят своей целью определить, что (какой блок кода и (или) данных) нужно изъять из ОП, чтобы освободить место для записи поступающих кодов и (или) данных.

## Типы адресов

Для идентификации переменных и команд на разных этапах жизненного цикла программы используются символьные имена, виртуальные адреса и физические адреса, как показано на рис. 2.


Идентификаторы переменных в программе на алгоритмическом языке

Условные адреса, вырабатываемые транслятором

Номера ячеек в ОП

2. Типы адресов

Символьные имена присваивает пользователь, создавая переменные, когда пишет программу на алгоритмическом языке.

Виртуальные адреса вырабатывает транслятор, который переводит программу на машинный язык. Поскольку во время трансляции обычно не

известно, в какое место ОП будет загружена программа, то транслятор присваивает переменным и командам виртуальные (условные) адреса, отсчитывая их от нулевого адреса. Физические адреса соответствуют номерам ячеек ОП, в которых будут размещаться переменные и команды этой программы.

**Виртуальным адресным пространством** процесса называется совокупность всех виртуальных адресов этого процесса. Диапазон возможных адресов виртуального пространства у всех процессов, работающих с одной и той же ОС, одинаков. Например, в 32-разрядной ОС этот диапазон задается границами 00000000 и FFFFFFFF (в шестнадцатеричной системе счисления).

## Алгоритмы распределения памяти

При распределении ОП между процессами

возникает множество вопросов, например: следует ли назначать каждому процессу одну непрерывную область в ОП или можно выделять адреса некоторыми пулами; должны ли части программы, загруженные в память, находиться на одном месте в течение всего периода выполнения процесса или можно их время от времени перемещать; что делать, если программа не помещается в имеющуюся свободную память?

Разные ОС по-разному отвечают на эти и другие вопросы по управлению памятью. Далее будут рассмотрены наиболее общие подходы к распределению памяти, которые были характерны для разных этапов истории ОС. Некоторые из них сохранили актуальность и используются сегодня, другие представляют только познавательный интерес.

На рис. 3 все алгоритмы распределения памяти

разделены на два основных класса: алгоритмы, в которых используется внешняя память для перемещения в нее сегментов данных и кода, и алгоритмы, в которых она не используется.

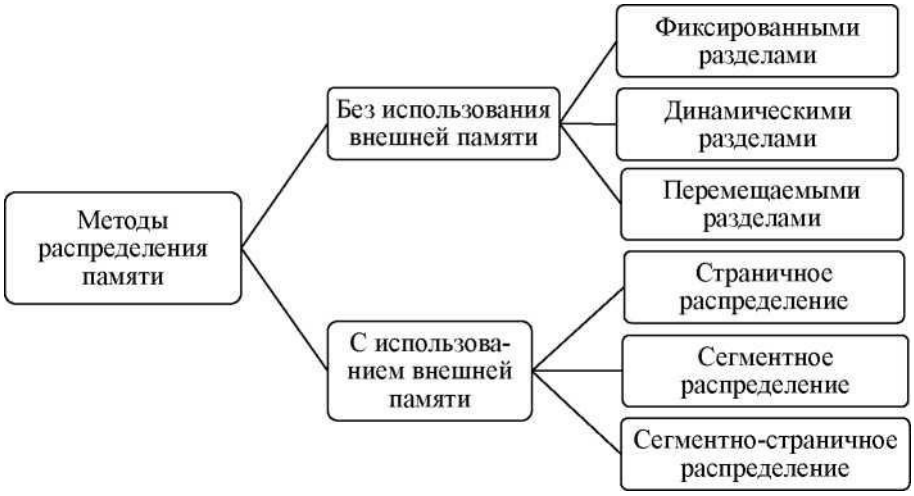



Рис. 3. Классификация методов распределения памяти

Наиболее простой способ управления ОП состоит в том, что память разбивается на несколько участков с фиксированным объемом, их называют разделами. Такое разбиение может быть выполнено вручную оператором (или самой

системой) во время старта системы. После этого границы разделов остаются постоянными. Очередная задача, поступившая на выполнение, помещается либо в общую очередь готовых процессов, либо в отдельную очередь к разделу.

При простоте реализации, которая является очевидным достоинством системы, она имеет существенный недостаток — отсутствие гибкости. В каждом разделе может выполняться только один процесс, поэтому уровень мультипрограммирования ограничен количеством созданных разделов. Еще один недостаток заключается в том, что невозможно выполнить процессы, программы которых не помещаются ни в один из разделов, но для которых было бы достаточно памяти нескольких разделов.

Такой способ управления памятью применялся в ранних мультипрограммных ОС. Однако и

сейчас метод распределения памяти фиксированными разделами находит применение в системах реального времени в основном благодаря небольшим затратам на реализацию. Детерминированность вычислительного процесса систем реального времени (заранее известен набор выполняемых задач, их требования к памяти, а иногда и моменты запуска) компенсирует недостаточную гибкость данного способа управления памятью.

В случае распределения памяти динамическими разделами, память машины не делится на разделы заранее. Каждой вновь поступившей на выполнение задаче на этапе создания процесса выделяется вся необходимая память (если такого объема памяти в данный момент нет, то задача не принимается на выполнение и процесс для нее не создается). По

завершении процесса память освобождается, и на это место может быть загружен другой процесс. Таким образом, в момент времени *t* память представляет собой случайную последовательность занятых и свободных участков (разделов) произвольного размера.

По сравнению с методом распределения памяти фиксированными разделами, данный метод обладает большей гибкостью, но имеет серьезный недостаток — склонность к фрагментации памяти. Под **фрагментацией** понимается наличие значительного числа несмежных между собой участков свободной памяти, которые имеют небольшой размер (это и есть фрагменты). Размер каждого из фрагментов недостаточен для того, чтобы хотя бы одна из готовых к выполнению программ могла поместиться в нем, при этом суммарный объем

фрагментов может значительно превышать потребности любой из этих программ.

Распределение памяти динамическими разделами лежит в основе подсистем управления памятью многих мультипрограммных ОС I960—70-х гг., например такой, как OS/360.

Одним из методов борьбы с фрагментацией является перемещение всех занятых участков в одном направлении таким образом, чтобы вся свободная память объединилась в одну свободную область. В результате этого у ОС появляется еще одна функция по управлению ОП — перемещение содержимого разделов из одного места памяти в другое с одновременной корректировкой имеющейся в системе таблицы свободных и занятых областей памяти. Эту процедуру называют сжатием, хотя чаще употребляется другой термин — «сбор мусора».

Хотя процедура сжатия и приводит к более эффективному использованию памяти, она значительно расходует ресурсы системы, что перевешивает ее достоинства.

## Свопинг и виртуальная память

(Свопинг — один из механизмов виртуальной памяти. При продолжительном неиспользовании оперативной памяти или какой-либо её части она перемещается на другой носитель информации (например, определённый участок жёсткого диска). Тем самым освобождается более быстродействующая оперативная память, необходимая для выполнения текущих задач. )

Программа должна быть загружена в ОП для того, чтобы она могла выполняться. Только тогда процессор сможет получать команды и выполнять определенные в них действия. Объем ОП компьютера сказывается на характере и скорости протекания вычислительного процесса. Он определяет число одновременно работающих процессов и размеры их адресных пространств.

Если необходимо выполнять одновременно

большое количество задач, требуется иметь достаточно большой объем ОП. В условиях, когда для обеспечения приемлемого уровня мультипрограммирования имеющейся ОП недостаточно, было предложено так организовать вычислительный процесс, чтобы данные и коды некоторых процессов целиком или частично временно выгружались на диск.

В режиме мультипрограммирования, наряду с активным (находящимся в состоянии выполнения) процессом, имеются процессы, находящиеся в состояниях ожидании или готовности (т. е. приостановленные процессы).

Коды и данные таких процессов могут быть временно вытеснены на диск. Несмотря на то что коды и данные процесса в какой-то момент времени отсутствуют в ОП, ОС знает о его существовании и учитывает это при планировании

процессорного времени и других системных ресурсов.

К тому моменту, когда процесс выбран для загрузки на процессор, его коды и данные возвращаются с диска в ОП. Если при этом оказывается, что свободного места в ОП не хватает, то на диск выгружаются данные и коды какого-то другого процесса.

Такая подмена (виртуализация) ОП дисковой памятью позволяет повысить уровень мультипрограммирования, при этом объем ОП компьютера не столь строго ограничивает количество одновременно выполняемых процессов, поскольку суммарный объем памяти, занятый этими процессами, может существенно превосходить объем ОП, имеющейся на данном ПК.

**Виртуальным** называется **ресурс**, который

пользователю или пользовательской программе представляется обладающим свойствами, которыми он в действительности не обладает.

В данном случае в распоряжение программиста предоставляется виртуальная ОП, размер которой значительно больше объема ОП, реально установленной в ПК.

Пользователь пишет программу, а транслятор, используя виртуальные адреса, переводит ее в машинные коды так, как будто в распоряжении программы имеется однородная ОП большого объема.

В реальности коды и данные, используемые программой, хранятся во вторичной, дисковой, памяти и только при необходимости загружаются в физическую ОП. В этом случае становится очевидным, что работа такой ОП происходит значительно медленнее.

Виртуализация ОП осуществляется совокупностью программных модулей ОС и аппаратных схем процессора и включает решение следующих задач:

– размещение данных в запоминающих устройствах разного типа, например, часть кодов программы в ОП, а часть — на диске;

– выбор образов процессов или их частей для перемещения из ОП на диск и обратно;

– перемещение по мере необходимости данных между ОП и диском;

– преобразование виртуальных адресов в физические.

Очень важно то, что все действия по организации взаимодействия диска и ОП осуществляются ОС и аппаратурой процессора без участия пользователя и никак не отражаются

на работе приложений.

Виртуализация памяти может быть осуществлена на основе двух различных подходов:

– свопинга, или подкачки — образы процессов выгружаются на диск и возвращаются в ОП целиком;

– виртуальной памяти *(virtual memory)* — между ОП и диском перемещаются части (сегменты, страницы) образов процессов.

Свопинг является частным случаем виртуальной памяти и представляет собой более простой способ совместного использования ОП и диска.

Однако этой технологии присуща избыточность: для выполнения процесса не нужно загружать в ОП все его коды и данные, вполне достаточно загрузить небольшую часть

кода, которая содержит нужную для выполнения инструкцию и те данные, которые использует эта инструкция, а также предоставить место для размещения сегмента стека.

Подобным же образом не требуется выгружать другой процесс на диск целиком для освобождения памяти — часто достаточно вытеснить на диск только часть его кода и данных.

Перемещение избыточных объемов информации замедляет работу всей системы и приводит к неэффективному использованию памяти. Более того, системы, использующие свопинг, имеют еще один недостаток: они не способны работать с процессами, виртуальные адресные пространства которых превышают имеющуюся в наличии свободную ОП.

Из-за указанных недостатков свопинг как основной механизм управления памятью не

используется в современных версиях ОС. На смену ему пришел более эффективный механизм виртуальной памяти, основная идея которого состоит в том, что при нехватке места в физической ОП, данные и коды процесса выгружаются на диск частично.

Основной проблемой систем виртуальной памяти является преобразование виртуальных адресов в физические, поскольку коды и данные процессов многократно меняют свое местоположение в ОП. Решение этой проблемы зависит от того, какой способ структуризации виртуального адресного пространства принят в данной системе управления памятью.

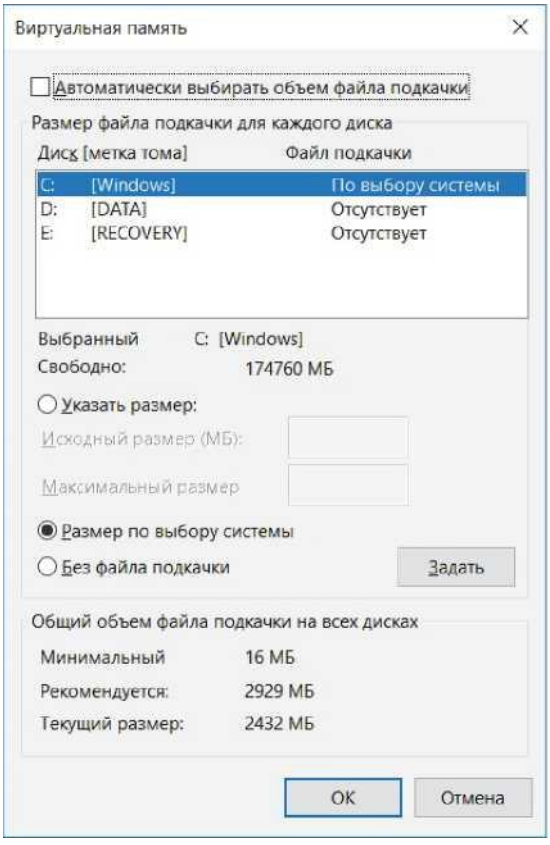
Реализацию систем виртуальной памяти можно разделить на три класса:

– страничная виртуальная память — организует перемещение данных между памятью

и диском страницами — частями виртуального адресного пространства, фиксированного и сравнительно небольшого размера;

– сегментная виртуальная память — предусматривает перемещение данных сегментами — частями виртуального адресного пространства произвольного размера, полученными с учетом смыслового значения данных;

– сегментно-страничная виртуальная память — использует двух-уровневое деление: виртуальное адресное




4. Окно конфигурирования виртуальной памяти в Windows 10

пространство делится на сегменты, а затем сегменты делятся на страницы. Единицей перемещения данных в этом классе является страница. Такой способ управления памятью объединяет в себе элементы двух первых классов.

Для хранения сегментов и страниц на диске отводится либо специальная область (дисковый раздел, часто называемый swap-разделом), либо специальный файл, который называют файлом подкачки. Другое популярное название этого файла — страничный файл *(page file,* или *paging file),* его текущий размер является важным системным параметром, который оказывает влияние на возможности ОС: чем он больше, тем больше процессов может одновременно выполняться в ОС (при фиксированном размере ОП). Однако необходимо понимать, что увеличение числа одновременно работающих процессов

за счет увеличения размера страничного файла замедляет их работу, т. к. значительная часть времени при этом тратится на перекачку кодов и данных из ОП на диск и обратно.

Пример окна конфигурирования параметров виртуальной памяти (в т. ч. файла подкачки) в системе Windows 10 показан на рис.4.