**Лекция 9. Управление памятью**

**Цель занятия:** сформировать знания студентов об управлении памятью и сегментной организации памяти.

**Управление памятью.**

Организация и управление основной памятью компьютера - один из самых важных факторов, определяющих построение ОС. Для непосредственного выполнения программ или обращения к данным необходимо, чтобы они размещались в основной памяти.

Вторичная, или внешняя память (жесткий диск), хранит множество программ и данных, которые должны быть наготове для обработки, т.е. загружены в основную память.

**Организация памяти.**

Под организацией памяти понимается, каким образом предоставляется и используется основная память, т.е. необходимо дать ответы на следующие вопросы:

1. Будем ли мы помещать в основную память только одну программу пользователя или несколько программ одновременно.
2. Если в основной памяти размещается программ сразу, будем ли мы предоставлять каждой из них одинаковое количество ячеек памяти или разобьем ее на части (разделы) различных размеров.
3. Будем ли разбивать основную память жестким образом, т.е. разделы определяются на достаточно длительное время, либо динамически, позволяя компьютеру быстро реагировать на изменения потребностей программы в ресурсах.
4. Будем ли мы требовать такого построения программы пользователя, чтобы они выполнялись только в конкретном разделе, либо предусмотрим возможность выполнения программы с занятием любых подходящих для них разделов.
5. Будем ли мы требовать, чтобы каждая программа помещалась в одном, непрерывном блоке ячеек памяти, либо допустим возможность разбиения программы на отдельные блоки, размещаемые в любых свободных участках (дырах) основной памяти.

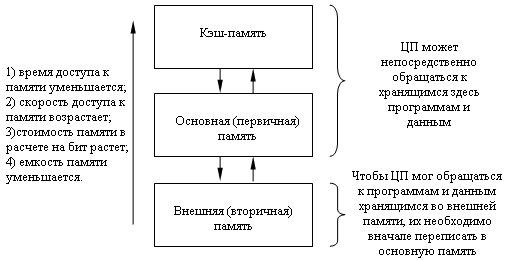
**Иерархия памяти.**

Чтобы программы можно было выполнять, они должны находиться в основной памяти, а программы и данные, которые в настоящий момент не нужны, можно хранить во внешней памяти.

Современные ОС поддерживают 3 уровня памяти:

* кэш-память
* основная
* внешняя

Иерархическая организация памяти:



Стратегии управления памятью.

Стратегии направлены на то, чтобы обеспечить наилучшее возможное использование ресурсов основной памяти.

Категории управления памятью.

1. Стратегии выборки. Ставят своей целью определить, когда следует "втолкнуть" очередной блок программы в основную память.
   * по запросу (по требованию). Очередной блок программы или данных загружается, когда его запрашивает работающая программа.
   * стратегии упреждающей выборки. ОС заранее предугадывает, какой блок программы или данных будет востребован, значит, он и будет заранее загружен в основную память.
2. Стратегии размещения. Целью является определить, в какое место основной памяти следует помещать поступающую программу. Используются стратегии размещения:
   * "первого подходящего"
   * "наиболее подходящего"
   * "наименее подходящего" - по размерам свободного участка памяти.
3. Стратегии замещения. Ставят целью определить, какой блок программы или данных следует вытеснить из основной памяти, чтобы освободить место для записи последующих программ и данных.

**Логическая организация памяти:**

При рассмотрении организации памяти следует иметь в виду, что рассматривается два отдельных вопроса:

1. Логическая организация памяти внутри выполняющегося процесса.
2. Физическая организация памяти , методы отображения памяти процесса на физическую память ЭВМ.

**Логическая организация памяти:**

Логическая память может быть организована 2 различными видами:

|  |  |
| --- | --- |
| 1. Непрерывная единая память   Image_html_m28d26aa5.gif | 1. Сегментная организация Imageпамяти. |

Два основных вида организации виртуальной памяти – **сегментная и страничная организация**.

**При сегментной организации** вся виртуальная память, используемая программой, разбивается на части, называемые **сегментами**. Это разбиение выполняется либо самим программистом (если он программирует на языке ассемблера), либо компилятором используемого языка программирования. Размеры сегментов могут быть различными, но в пределах максимального размера, используемого в данной архитектуре. Разбиение обычно производится  на логически осмысленные части, такие, как сегмент данных, сегмент кода, сегмент стека и т.п. Большая программа может содержать несколько сегментов одного типа, например, несколько сегментов кода или данных.

Таким образом, при сегментной организации у программы нет единого линейного адресного пространства. Виртуальный адрес состоит из двух частей: **селектора сегмента** и **смещения** от начала сегмента.

Селектор сегмента представляет некоторое число, которое обычно является индексом в таблице сегментов данного процесса. Такая таблица содержит для каждого сегмента его размер, режим доступа (только чтение или возможна запись), флаг присутствия сегмента в памяти. Если сегмент находится в памяти, то в таблице хранится его базовый адрес (адрес физической памяти, соответствующий началу сегмента). Отсутствие сегмента означает, что его данные временно вытеснены на диск и хранятся в **файле подкачки** (swap file).

В кодах программы селектор сегмента может либо явно присутствовать как часть адреса, либо подразумеваться в зависимости от смысла конкретного адреса. Например, для адресов выполняемых команд используется селектор текущего сегмента команд, а для адресов операндов – селектор текущего сегмента данных.

При каждом обращении к виртуальному адресу аппаратными средствами выполняется преобразование пары «сегмент : смещение» в физический адрес. Упрощенная схема такого преобразования показана на рис. 1.

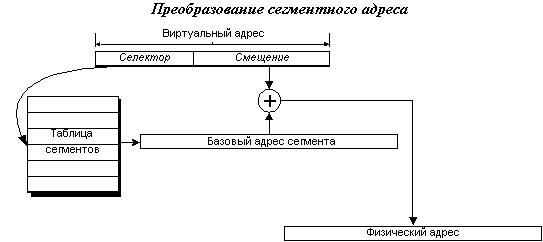


Рис. 1

Селектор сегмента используется для доступа к соответствующей записи таблицы сегментов. Если данный сегмент присутствует в памяти, то его базовый адрес, прочитанный в таблице, складывается со смещением из виртуального адреса. Результат сложения представляет собой физический адрес, по которому и происходит обращение к памяти.

Если сегмент отсутствует в памяти, то происходит прерывание. Обрабатывая его, система должна подгрузить сегмент с диска на свободное место в памяти, записать его базовый адрес в таблицу сегментов и затем повторить команду, вызвавшую прерывание.

Но откуда возьмется свободное место в памяти? По всей вероятности, системе придется для этого убрать из памяти какой-то другой сегмент, принадлежащий либо к этому же, либо к иному процессу. Копия вытесняемого сегмента должна остаться в файле подкачки. Чтобы избежать лишней работы, в каждой записи таблицы хранится флаг, отмечающий, является ли сегмент в памяти «чистым» или «грязным», т.е. совпадает ли его содержимое с дисковой копией или же оно было изменено в памяти после последней загрузки с диска. «Грязный» сегмент должен быть сохранен на диске, для «чистого» сохранение не требуется. Если сегмент определен как доступный только для чтения, то он заведомо «чистый».

Поскольку сегменты имеют различные размеры, то в ходе работы системы, сопровождающейся многократной загрузкой и выгрузкой сегментов, возникает эффект фрагментации памяти. Во всех случаях причиной фрагментации является многократное занятие и освобождение областей различного размера.

Для борьбы с фрагментацией можно время от времени производить дефрагментацию, т.е. перемещение всех сегментов, находящихся в памяти, на новые места, без «дырок» в памяти между сегментами. При этом, однако, требуется, чтобы система откорректировала таблицы сегментов всех тех процессов, сегменты которых переместились в физической памяти. Кроме того, перемещение сегментов занимает ощутимое время, поэтому оно недопустимо для сегментов, содержащих, например, обработчики прерываний, которые должны срабатывать очень быстро. Чтобы избежать этих проблем, в некоторых системах сегменты могут находиться в одном из двух состояний:

· **фиксированный** сегмент не должен перемещаться в памяти;

· **перемещаемый** сегмент может перемещаться системой, однако программа не может обращаться к адресам в таком сегменте, поскольку его местоположение не определено.

Чтобы работать с данными в перемещаемом сегменте, программа должна предварительно временно зафиксировать его вызовом специальной системной функции. При этом ОС определяет текущее местоположение сегмента и корректирует его базовый адрес в таблице сегментов процесса. Если же программа в течение некоторого времени не планирует обращаться к данному сегменту, то его следует расфиксировать, поскольку чем больше фиксированных сегментов в системе, тем менее эффективна будет дефрагментация.

При переключении текущего процесса все, что должна сделать система в отношении памяти, заключается в замене таблицы сегментов. Для этого либо в специальный системный регистр записывается адрес таблицы сегментов текущего процесса, либо, если аппаратура допускает только одну таблицу сегментов, ее содержимое должно быть перезаписано так, чтобы соответствовать новому работающему процессу.

**Контрольные вопросы:**

* 1. *Какие существуют типы организации памяти?*
  2. *Что такое сегментная организация памяти и как она происходит?*