УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные понятия, дайте ответы на контрольные вопросы.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 22.05.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция
Тема: Виртуальная и физическая память

Оперативная память является, пожалуй, одним из наиболее дорогих компонентов компьютерной системы. Ранние системы UNIX имели в своем распоряжении 64 Кбайт оперативной памяти, и это количество было явно недостаточным, современные компьютеры обладают гигабайтами оперативной памяти, но и этого уже мало.

Оперативная память может быть представлена в виде последовательности байтов, каждый из которых имеет свой уникальный адрес, называемый *физическим адресом*. Именно эти адреса в конечном счете использует процессор, обмениваясь данными с оперативной памятью. Однако адресное пространство процесса существенным образом отличается от адресного пространства физической оперативной памяти. Представим себе, что адресное пространство процесса непосредственно отображалось бы в оперативную память, другими словами, что адреса, используемые процессом, являлись бы физическими адресами. При таком подходе на пути создания многозадачной системы нас ожидал бы ряд непреодолимых препятствий:

- Во-первых, трудно себе представить механизм, защищающий адресное пространство одного процесса, от адресного пространства другого или, что более важно, от адресного пространства самой операционной системы. Поскольку каждый процесс работает с физическими адресами, нет никакой

гарантии, что процесс не обратится к ячейкам памяти, принадлежащим другим процессам или ядру системы. Последствия такого обращения скорее всего будут весьма плачевными.

- Во-вторых, уже на этапе компиляции необходимо было бы предусмотреть распределение существующего физического адресного пространства. При запуске каждый процесс должен занимать непрерывную и непересекающуюся область физических адресов.

- В-третьих, подобное распределение памяти между процессами вряд ли можно назвать оптимальным. Объем физической оперативной памяти будет существенным образом ограничивать число процессов, одновременно выполняющихся в системе. Так восемь процессов, каждый из которых занимает 1 Мбайт памяти, исчерпают 8 Мбайт оперативной памяти, а операционная система при средней загрузке насчитывает более 80 процессов!

Все перечисленные проблемы преодолимы с помощью виртуальной памяти. При этом адреса, используемые приложениями и самим ядром, не обязаны соответствовать физическим адресам. Виртуальные адреса транслируются или отображаются в физические на аппаратном уровне при активном участии ядра операционной системы.

Смысл виртуальной памяти заключается в том, что каждый процесс выполняется в собственном *виртуальном адресном пространстве*. Виртуальное адресное пространство — настоящий рай для процесса. Во-первых, у процесса создается ощущение исключительности — ведь все адресное пространство принадлежит только ему. Во-вторых, он больше не ограничен объемом физической памяти — виртуальная память может значительно превышать физическую. В результате процессы становятся изолированными друг от друга и не имеют возможности (даже при желании) "хозяйничать" в адресном пространстве соседа. Физическая память распределяется максимально эффективно — она не зависит от распределения виртуальной памяти отдельного процесса.

Очевидно, что для реализации виртуальной памяти необходим управляемый механизм отображения виртуального адреса в физический. В

современных компьютерных системах процесс отображения выполняется на аппаратном уровне (с помощью обеспечивая высокую скорость трансляции. Операционная система осуществляет управление этим процессом.

Современные процессоры, как правило, поддерживают объединение адресного пространства в области переменного размера — *сегменты* и области фиксированного размера — *страницы*. При этом для каждого сегмента или страницы может быть задано собственное отображение виртуальных адресов в физические.

На рис. 1 показана взаимосвязь между виртуальным и физическим адресным пространством. Виртуальное адресное пространство процесса, как правило, является последовательным в рамках уже знакомых нам сегментов — кода, данных, стека и библиотек. Расположение соответствующих областей физической памяти может иметь фрагментированный характер, позволяя оптимально распределять память между процессами.

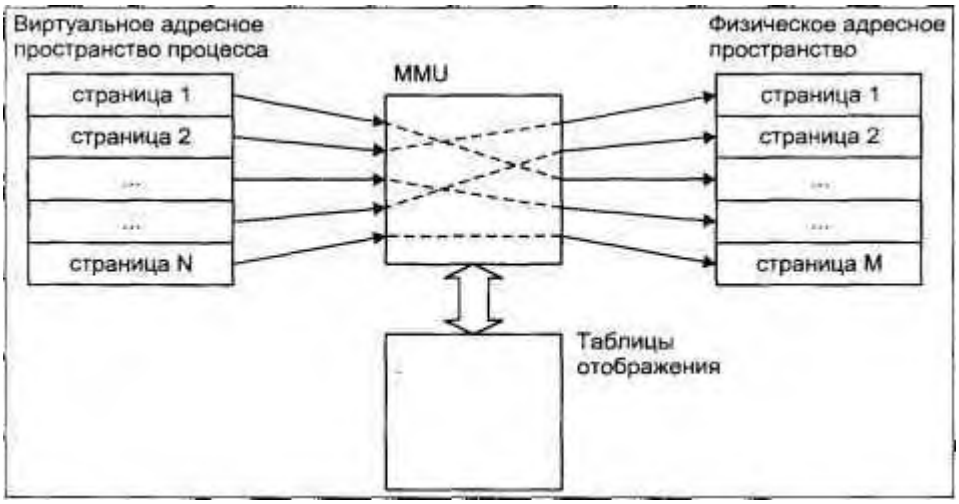

Рис. 1. Виртуальная и физическая память

Размер виртуальной памяти может существенно превышать размер физической за счет использования *вторичной памяти* или *области свопинга* — как правило, дискового пространства, где могут сохраняться временно не используемые участки адресного пространства процесса. Например, если при выполнении процесса происходит обращение к виртуальному адресу, для которого присутствует соответствующая страница физической памяти, операция чтения или записи завершится успешно. Если страница в оперативной памяти отсутствует, процессор генерирует аппаратное прерывание, называемое

страничной ошибкой (page fault), в ответ на которое ядро определяет положение сохраненного содержимого страницы в области свопинга, считывает страницу в память, устанавливает параметры отображения виртуальных адресов в физические и сообщает процессору о необходимости повторить операцию. Все эти действия невидимы для приложения, которое работает с виртуальной памятью.

Механизм отображения виртуальных адресов в физические (трансляция адреса) существенным образом зависит от конкретной аппаратной реализации. Чтобы наше обсуждение не носило слишком абстрактного характера, в этом разделе рассмотрим механизм отображения виртуальных адресов в физические в операционной системе SCO UNIX на примере семейства процессоров Intel. Однако, как и для остальных подсистем UNIX, основные принципы отличаются мало, и данное изложение поможет читателю представить механизмы управления памятью и разобраться, при необходимости, в конкретной реализации.