**БИЛЕТ № 12**

1) Процедура начальной загрузки ОС.

Вначале происходит тестирование системы – определяются параметры системы.
Загрузка с BIOS – программа начальной загрузки – загружает частично таблицу векторов прерываний в ОП, и переписывает программы для этой таблицы. Загрузка с Boot-сектора бутового загрузчика, который определяет активный раздел в соответствии находится начало активного раздела. Бутовый загрузчик находится в MBR. По адресу 0 0 0 находится jump на программу начальной загрузки в активном разделе. Загрузка ОС(инициализация) происходит в два этапа: init-ядра, init-sys
Когда синициализировали систему, подгружаем autoexec и command.com

2) Система управления памятью. Решаемые задачи.

Функции системы управления памятью
Чтобы обеспечить эффективный контроль использования памяти, ОС должна выполнять следующие функции:

- отображение адресного пространства процесса на конкретные области физической памяти;
- распределение памяти между конкурирующими процессами;
- контроль доступа к адресным пространствам процессов;
- выгрузка процессов (целиком или частично) во внешнюю память, когда в оперативной памяти недостаточно места;
- учет свободной и занятой памяти.

В следующих разделах лекции рассматривается ряд конкретных схем управления памятью. Каждая схема включает в себя определенную идеологию управления, а также алгоритмы и структуры данных и зависит от архитектурных особенностей используемой системы. Вначале будут рассмотрены простейшие схемы. Доминирующая на сегодня схема виртуальной памяти будет описана в последующих лекциях.
Простейшие схемы управления памятью
Первые ОС применяли очень простые методы управления памятью. Вначале каждый процесс пользователя должен был полностью поместиться в основной памяти, занимать непрерывную область памяти, а система принимала к обслуживанию дополнительные пользовательские процессы до тех пор, пока все они одновременно помещались в основной памяти. Затем появился "простой свопинг" (система по-прежнему размещает каждый процесс в основной памяти целиком, но иногда на основании некоторого критерия целиком сбрасывает образ некоторого процесса из основной памяти во внешнюю и заменяет его в основной памяти образом другого процесса). Такого рода схемы имеют не только историческую ценность. В настоящее время они применяются в учебных и научно-исследовательских модельных ОС, а также в ОС для встроенных (embedded) компьютеров.

3) Особенности управления внешней памятью в системах UNIX.

Модуль распределения памяти контролирует выделение памяти процессам. Если в какой-то момент система испытывает недостаток в физической памяти для запуска всех процессов, ядро пересылает процессы между основной и внешней памятью с тем, чтобы все процессы имели возможность выполняться. Существует два способа управления распределением памяти: выгрузка (подкачка) и замещение страниц. Программу подкачки иногда называют планировщиком, т.к. она "планирует" выделение памяти процессам и оказывает влияние на работу планировщика центрального процессора.

Модуль "планировщик" распределяет между процессами время центрального процессора. Он планирует очередность выполнения процессов до тех пор, пока они добровольно не освободят центральный процессор, дождавшись выделения ресурса, или до тех пор, пока ядро системы не выгрузит их после того, как их время выполнения превысит заранее определенный квант времени. Планировщик выбирает на выполнение готовый к запуску процесс с наивысшим приоритетом; выполнение предыдущего процесса (приостановленного) будет продолжено тогда, когда его приоритет будет наивысшим среди приоритетов всех готовых к запуску процессов. Существует несколько форм взаимодействия процессов между собой: от асинхронного обмена сигналами о событиях до синхронного обмена сообщениями.

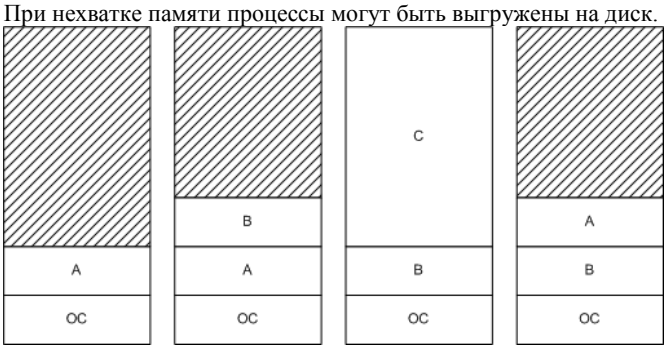
//

Так как памяти, как правило, не хватает. Для выполнения процессов часто приходится использовать диск.

Основные способы использования диска:
Свопинг (подкачка) - процесс целиком загружается в память для работы
Виртуальная память - процесс может быть частично загружен в память для работы

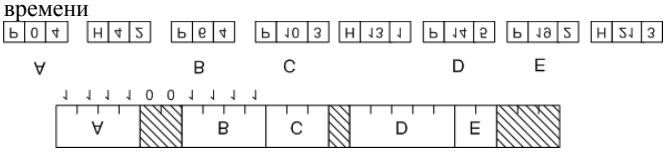
6.3.1 Свопинг (подкачка)

При нехватке памяти процессы могут быть выгружены на диск.




т.к. процесс C очень большой, процесс A был выгружен временно на диск,
после завершения процесса C он снова был загружен в память.
Как мы видим процесс A второй раз загрузился в другое адресное пространство, должны создаваться такие условия, которые не повлияют на работу процесса.
Свопер - планировщик, управляющий перемещением данных между памятью и диском.
Этот метод был основным для UNIX до версии 3BSD.

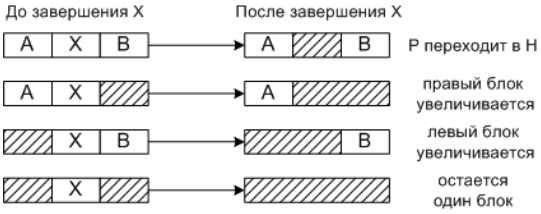
Управление памятью с помощью битовых массивов
Вся память разбивается на блоки (например, по 32бита), массив содержит 1 или 0 (занят или незанят). Чтобы процессу в 32Кбита занять память, нужно набрать последовательность из 1000 свободных блоков. Такой алгоритм займет много времени




битовые массивы и списки

Управление памятью с помощью связных списков
Этот способ отслеживает списки занятых (между процессами) и свободных (процессы) фрагментов памяти. Запись в списке указывает на:

- занят (P) или незанят (H) фрагмент
- адрес начала фрагмента
- длину фрагмента



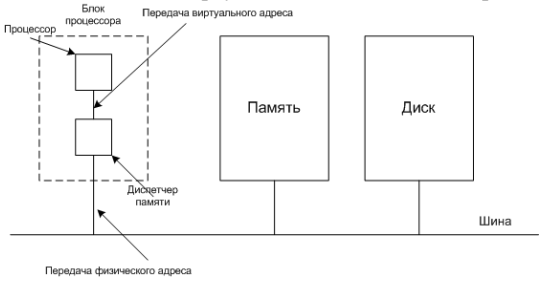

Четыре комбинации соседей для завершения процесса X

Алгоритмы выделения блока памяти:

- первый подходящий участок.
- следующий подходящий участок, стартует не сначала списка, а с того места на котором остановился в последний раз.
- самый подходящий участок (медленнее, но лучше использует память).
- самый неподходящий участок, расчет делается на то, что программа займет самый большой участок, а лишнее будет отделено в новый участок, и он будет достаточно большой для другой программы.

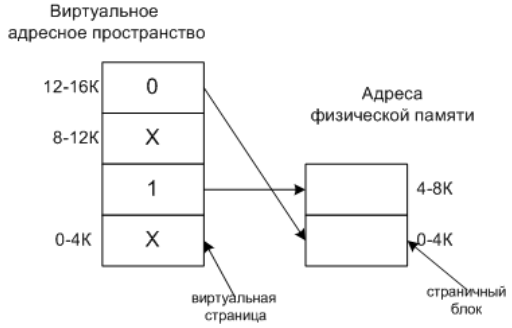
6.3.2 Виртуальная память

Основная идея заключается в разбиении программы на части, и в память эти части загружаются по очереди. Программа при этом общается с виртуальной памятью, а не с физической.




Диспетчер памяти преобразует виртуальные адреса в физические.

Страничная организация памяти
Страницы - это части, на которые разбивается пространство виртуальных адресов. Страничные блоки - единицы физической памяти. Страницы всегда имеют фиксированный размер. Передача данных между ОЗУ и диском всегда происходит в страницах.




X - обозначает не отображаемую страницу в физической памяти.
Страничное прерывание - происходит, если процесс обратился к странице, которая не загружена в ОЗУ (т.е. X). Процессор передается другому процессу, и параллельно страница загружается в память.
Таблица страниц - используется для хранения соответствия адресов виртуальной страницы и страничного блока.
Таблица может быть размещена:

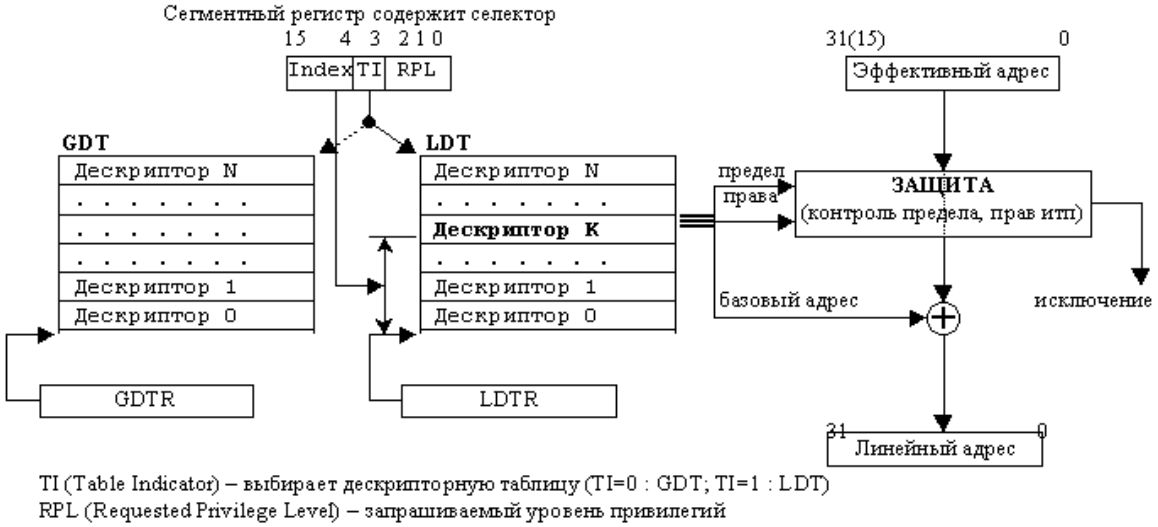
- в аппаратных регистрах (преимущество: более высокое быстродействие, недостаток - стоимость)
- в ОЗУ

Типичная запись в таблице страниц

Присутствие/отсутствие - загружена или незагружена в память Защита - виды доступа, например, чтение/запись. Изменение - изменилась ли страница, если да то при выгрузке записывается на диск, если нет, просто уничтожается. Обращение - было ли обращение к странице, если нет, то это лучший кандидат на освобождение памяти. Информация о адресе страницы когда она хранится на диске, в таблице не размещается. Для ускорения доступа к страницам в диспетчере памяти создают буфер быстрого преобразования адреса, в котором хранится информация о наиболее часто используемых страниц.
Страничная организация памяти используется, и в UNIX, и в Windows.

4) Состав системных объектов. Реализация защищенного режима.
Ответ:

Доп.инфа:
**Особенности адресации в защищенном режиме.**



TI (Table Indicator) – выбирает дескрипторную таблицу (TI=0 : GDT; TI=1 : LDT)
RPL (Requested Privilege Level) – запрашиваемый уровень привилегий

Дескриптор - это 8-байтная единица описательной информации, распознаваемая устройством управления памятью в защищенном режиме, хранящаяся в дескрипторной таблице. Дескриптор сегмента содержит базовый адрес описываемого сегмента, предел сегмента и права доступа к сегменту. Дескрипторы являются основой защиты и мультизадачности. В защищенном режиме сегменты могут начинаться с любого линейного адреса и иметь любой предел вплоть до 4Гбайт. Существуют две обязательных дескрипторных таблицы - глобальная (GDT) и дескрипторная таблица прерывания (IDT),- а также множество (до 8192) локальных дескрипторных таблиц (LDT), из которых в один момент времени процессору доступна только одна. Дескрипторы сегментов могут находиться в GDT или LDT. Расположение дескрипторных таблиц определяется регистрами процессора GDTR, IDTR, LDTR. Регистры GDTR и IDTR - 6-байтные, они содержат 32 бита линейного базового адреса дескрипторной таблицы и 16 бит предела таблицы. Программно доступная часть регистра LDTR - 16 бит, которые являются селектором LDT. Дескрипторы LDT находятся в GDT. Однако чтобы не обращаться каждый раз к GDT в процессоре имеется теневая (программно недоступная) часть регистра LDTR, в которую процессор помещает дескриптор LDT при каждой перегрузке селектора в регистре LDTR.

Значения, помещаемые в сегментные регистры, называются селекторами. Селектор содержит индекс дескриптора в дескрипторной таблице, бит определяющий, к какой дескрипторной таблице производится обращение (LDT или GDT), а также запрашиваемые права доступа к сегменту. Таким образом, селектор выбирает дескрипторную таблицу, выбирает дескриптор из таблицы, а по дескриптору определяется положение сегмента в линейном пространстве памяти. Однако обращение к дескрипторным таблицам происходит только при загрузке селектора в сегментный регистр. При этом процессор помещает дескриптор в теневую (программно недоступную) часть сегментного регистра. При формировании линейного адреса дескриптор сегмента процессору уже известен.

5) Сокеты, особенности их использования.
Soket – новая точка коммутации
КОНСПЕКТ тоже…
Сокеты – это средства межпроцессорного взаимодействия между процессорами разных вычислительных систем
Для обозначения коммуникационного узла, обеспечивающего прием и передачу данных для объекта (процесса), был предложен специальный объект — *сокет* (socket). Сокеты создаются в рамках определенного коммуникационного домена, подобно тому, как файлы создаются в рамках файловой системы. Сокеты имеют соответствующий интерфейс доступа в файловой системе, и так же как обычные файлы, адресуются некоторым целым числом — дескриптором. Однако в отличие от обычных файлов, сокеты представляют собой виртуальный объект, который существует, пока на него ссылается хотя бы один из процессов.

- *Сокет датаграмм* (datagram socket), через который осуществляется теоретически ненадежная, несвязная передача пакетов.
- *Сокет потока* (stream socket), через который осуществляется надежная передача потока байтов без сохранения границ сообщений. Этот тип сокетов поддерживает передачу экстренных данных.
- *Сокет пакетов* (packet socket), через который осуществляется надежная последовательная передача данных без дублирования с предварительным установлением связи. При этом сохраняются границы сообщений.
- *Сокет низкого уровня (raw* socket), через который осуществляется непосредственный доступ к коммуникационному протоколу.

Наконец, для того чтобы независимые процессы имели возможность взаимодействовать друг с другом, для сокетов должно быть определено *пространство имен.* Имя сокета имеет смысл только в рамках коммуникационного домена, в котором он создан.

Примитивы сокетов для TCP протокола:

- SOCKET – создать точку коммуникации
- BIND – назначить сокету локальный адрес
- LISTEN – обозначить готовность к установке соединения
- ACCEPT – заблокировать вызывающую сторону до прибытия запроса на соединение
- CONNECT – попытка установить соединение
- SEND, WRITETO, SENDTO – послать сообщение сокету в зависимости от типа сокета и сообщения
- RECEIVE, READ, RECIEVEFROM – принять сообщение

CLOSE – разорвать соединение

Коммуникационный канал создаётся при создании сокета между источником и получателем и имеет такие характеристики:

- Коммуникационный протокол
- Локальный адрес источника
- Локальный адрес процесса источника
- Удалённый адрес получателя

Удалённый адрес процесса получателя