Лекция 13:
Диски
Файловые системы

Алексей Линёв
Александр Мощук
Кирилл Погорельский

somes slides are adapted from the OS course at the University of Washington

Прогресс жестких дисков

- Емкость жестких дисков, 1975-1989 г.г.
  - удваивается каждые 3+ года
  - ежегодный прирост – 25%
  - за 10 лет – увеличение в 10 раз
  - экспоненциальный рост, но далеко не такой быстрый, как рост производительности процессоров
- Емкость жестких дисков, начиная с 1990 г.
  - удваивается каждые 12 месяцев
  - ежегодный прирост – 100%
  - за 10 лет – увеличение в 1000 раз
  - рост емкости жестких дисков опережает рост производительности процессоров в 10 раз!

Внешняя память (secondary storage)

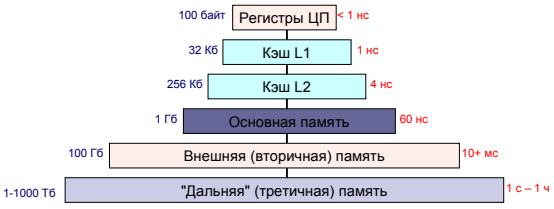
- Внешняя память, как правило:
  - все, что находится за пределами "основной памяти"
  - не позволяет непосредственно выполнять код, располагающийся в ней, или получать доступ к данным, используя адреса в инструкциях чтения/записи
- Характеристики
  - объемная: 50-1000 Гб
  - дешевая: $0.45/Гб
  - постоянная: отключение питания не приводит к потере данных
  - медленная: время доступа – миллисекунды
    - это медленно? почему?
  - подвержена сбоям, хоть и нечастым

- Буквально несколько лет назад стоимость диска рассчитывалась исходя их его объема в мегабайтах (и это была наша боль!)
- Сейчас 1 Гб (триллион байт) стоит $0.50
  - 1 Тб стоит $500, 1 Пб стоит $500К
- Через 2 года 1 Гб будет стоить $0.10
  - 1 Тб будет стоить $100, 1 Пб – $100К

Историческая справка …

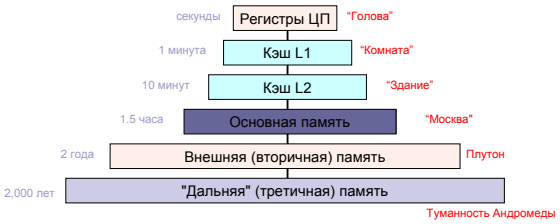
IBM 2314
Размер ~6 холодильников
8 x 29 Мб

Иерархия памяти



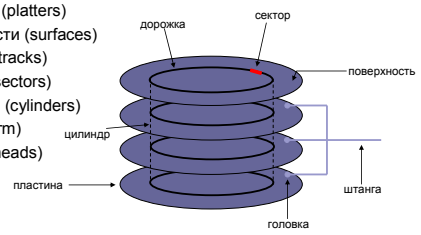
- Каждый уровень служит кэшем для нижележащих уровней

Иерархия памяти: "удаленность" данных

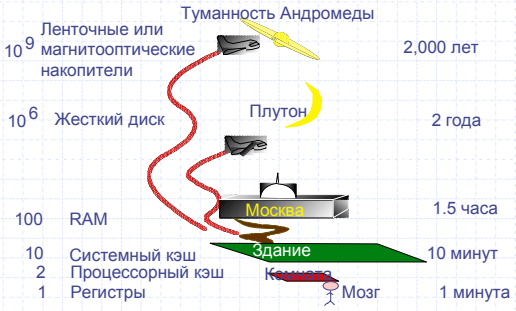


Физическая структура жесткого диска

- Компоненты жесткого диска
  - пластины (platters)
  - поверхности (surfaces)
  - дорожки (tracks)
  - сектора (sectors)
  - цилиндры (cylinders)
  - штанга (arm)
  - головки (heads)



Латентность устройств хранения: Насколько далеки ваши данные?



Производительность жестких дисков

- Время обработки запроса складывается из нескольких компонент
  - поиск (seek): перемещение штанги с головками к нужному цилиндру
    - зависит от скорости перемещения штанги
      - время поиска сокращается не слишком быстро (почему?)
  - вращение (латентность) (rotation/latency): ожидание, пока сектор не пройдет под головкой
    - зависит от скорости вращения диска
      - скорость вращения увеличивается, но медленно (почему?)
  - передача (transfer): передача данных с поверхности платина в контроллер диска, и затем из контроллера – получателю
    - зависит от плотности размещения байт на диске
      - увеличивается, причем очень быстро
- ОС при обращениях к диску пытается минимизировать все составляющие
  - в особенности поиск и вращение

Жесткие диски и ОС

- Жесткие диски – неоднородные устройства с нестабильными характеристиками
  - ошибки, испорченные сектора, промахи при поиске и т.д.
- Задачи ОС – скрыть эту неоднородность от высокоуровневых приложений
  - драйвера устройств (инициируют операции чтения и пр.)
  - абстракции более высокого уровня (файлы, базы данных,…)
- ОС может предоставлять различные уровни доступа к диску различным клиентам
  - физический сектор диска (Cylinder, Head, Sector)
  - логический блок диска (№ логического блока)
  - запись в файле (имя файла, № блока, записи или байта)

Планирование дисковых операций

- Поиск (seek) – очень длительная операция, и ОС пытается спланировать (переупорядочить) последовательность запросов к диску, ожидающих обслуживания
  - FCFS (first come, first serve) – не применять планирование
    - приемлемо при низкой загрузке
    - большое время ожидания при большой очереди запросов
  - SSTF (shortest seek time first) – первым обслуживается запрос с минимальным временем поиска
    - минимизирует движение штанги (поиск), достигается максимальная пропускная способность
    - благоволит к блокам в середине диска
  - SCAN (elevator algorithm) – алгоритм "лифта"
    - обслуживаем запросы, требующие продолжения поиска в одном направлении, затем меняем направление поиска
    - среднее время поиска распределено неравномерно (почему?)
  - C-SCAN
    - принцип, как в SCAN, но поиск производится всегда в одном направлении (пишущая машинка)
    - обеспечивает одинаковое время ожидания для всех запросов

Взаимодействие с дисками

- В старые добрые времена…
  - ОС должна была указывать № цилиндра, № головки, № сектора, объем передачи
    - то есть, ОС должна была знать все параметры диска
- Современные жесткие диски сами по себе очень сложны
  - не все дорожки имеют одинаковый размер, поддерживается переназначение секторов
  - диск предоставляет интерфейс высокого уровня, например SCSI
    - представляет данные диска в виде массива блоков [0 … N]
    - отображает адреса логических блоков в адреса в виде <цилиндр:головка:сектор>
    - ОС должна просто указать № логического блока, жесткий диск сам отображает его на триаду <цилиндр:головка:сектор>
    - непосредственно на диске присутствует кэш
    - в итоге, физические параметры скрыты от ОС
      - это и хорошо, и плохо

Основные операции

Unix

- create(name)
- open(name, mode)
- read(fd, buf, len)
- write(fd, buf, len)
- sync(fd)
- seek(fd, pos)
- close(fd)
- unlink(name)
- rename(old, new)

NT

- CreateFile(name, CREATE)
- CreateFile(name, OPEN)
- ReadFile(handle, …)
- WriteFile(handle, …)
- FlushFileBuffers(handle, …)
- SetFilePointer(handle, …)
- CloseHandle(handle, …)
- DeleteFile(name)
- CopyFile(name)
- MoveFile(name)

Файловые системы

- Идея файловых систем достаточно проста
  - это абстракция для работы со внешней памятью
    - часть этой абстракции – объект типа "файл"
  - логическая организация файлов в каталоги (директории)
    - иерархия каталогов (директорий)
  - разделение данных между процессами, пользователями и компьютерами
    - целостность, контроль доступа,…

Методы доступа к данным файла

- Некоторые файловые системы предоставляют различные методы доступа, определяющие порядок доступа приложений к данным
  - sequential access (последовательный доступ)
    - читаем по порядку байт за байтом
  - direct access (прямой доступ)
    - доступ по произвольному адресу в виде № блока/байта
  - record access (доступ по записям)
    - содержимое файла представляет собой массив записей постоянной или переменной длины
  - indexed access (доступ с использованием индексов)
    - ФС поддерживает индексирование всех записей файла по какому-либо полю
    - приложение может найти файл по значению этой записи (как в БД)
- Почему мы должны различать последовательный и прямой доступ?
  - чем должно отличаться функционирование ФС в этих случаях?

Файлы

- Файл – именованный набор данных, имеющий ряд характеристик
  - содержимое, размер, владелец, время последнего использования, права доступа,…
- Файлы могут быть различных типов
  - различие может быть на уровне файловой системы
    - файл, каталог, файл устройства, мягкая ссылка,…
  - или на уровне других компонент ОС или библиотек
    - двоичный исполняемый файл, dll, исходный текст программы, объектный файл, текстовый файл,…
- Тип файла может быть закодирован в его имени или содержимом
  - в Windows принято указывать тип файла в его имени
    - .com, .exe, .bat, .dll, .jpg, .mov, .mp3, …
  - старые версии Mac OS хранили вместе с файлом имя программы, создавшей его
  - в UNIX есть немного и того и другого
    - содержимое файла может содержать "магические числа" или начальные символы (например, #!)

Каталоги (директории)

- Каталоги (директории) обеспечивают
  - возможность пользователям упорядочить свои файлы
  - способ именования файлов, приемлемый как для пользователей, так и для ФС
- Большинство типов файловых систем поддерживают вложенность каталогов
  - иерархические имена (/, /usr, /usr/local, /usr/local/bin, …)
- В большинстве систем управления файлами (СУФ) поддерживается понятие "текущего каталога"
  - абсолютное (полное) имя: полностью указанный путь к файлу, начиная от корня ФС

    ```
    bash$ cd /usr/local
    ```

  - относительное имя: путь относительно текущего каталога

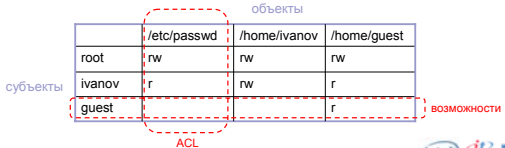
    ```
    bash$ cd /usr/local   (абсолютное)
    bash$ cd bin          (относительное, здесь эквивалентно cd /usr/local/bin)
    ```

Внутренняя структура каталога

- Каталог – это часто просто файл, содержащий специальные метаданные
  - каталог = перечень (имя файла, атрибуты файла)
  - атрибуты включают в себя
    - размер, права доступа, размещение данных на диске, время создания, время последнего использования,…
  - перечень файлов в каталоге обычно неупорядочен
    - когда вы набираете "ls", программа "ls" сортирует файлы для вас

Модель представления атрибутов защиты

- Два различных подхода
  - списки управления доступом (access control lists, ACLs)
    - для каждого объекта поддерживается список субъектов и разрешенных для них операций
  - перечни возможностей (capabilities)
    - для каждого субъекта поддерживается список объектов и разрешенных данному субъекту операций над ними
- Оба подхода могут быть описаны следующей матрицей



Трансляция имени файла

- Допустим, вы хотите открыть файл “/one/two/three”
  - `fd = open("/one/two/three", O_RDWR);`
- Что происходит внутри системы управления файлами?
  - открываем каталог "/" (всегда может быть найден)
  - ищем в перечне каталога "one", получаем информацию о размещении "one"
  - открываем каталог "one", ищем "two", получаем информацию о размещении "two"
  - открываем каталог "two", ищем "three", получаем информацию о размещении "three"
  - открываем файл "three"
  - (естественно, на каждом шаге проверяются права доступа)
- Система управления файлами тратит много времени на трансляцию имен файлов
  - поэтому операция открытия файла "open" отделена от операций чтения-записи (и поддерживаются контексты файлового ввода-вывода)
  - ОС обычно кэширует результаты поиска в префиксах имен файлов для повышения производительности
    - в именах "/a/b", "/a/bb", "/a/bbb" общий префикс “/a”

ACLs vs. Перечни возможностей

- Перечни возможностей проще передавать
  - как ключи – можно кому-либо передать
  - упрощают разделение доступа к объектам
- ACL более легко управляемые
  - привязаны к объектам, их проще предоставлять и отменять
    - для отмены возможности, сначала необходимо найти субъекта, владеющего ей
    - а это достаточно трудно сделать, учитывая возможность передачи возможностей между пользователями
- ACL становятся очень большими для объектов, к которым имеют доступ многие субъекты
  - можно упростить, введя понятие "группы пользователей"
    - пользователи включаются в группы, в ACL группам назначаются права
    - например, в Win2k вы наверняка входите в группу "Пользователи", или "Опытные пользователи" или "Администраторы"
  - дополнительное преимущество
    - изменение членства субъекта в группе означает изменение его прав доступа ко всем объектам, в ACL которых упоминается данная группа

Защита файлов

- СУФ должна каким-либо образом реализовывать защиту файлов
  - для управления тем, кто имеет доступ к файлу (пользователь)
  - для управления тем, какие операции они могут выполнять с файлом (чтение, запись, выполнение,…)
- В более общем смысле
  - понятие "файл" обобщается до понятия "объект“ (object)
  - понятие "пользователь" обобщается до понятия "субъект“ (principal)
  - понятия "чтение/запись" обобщаются до понятия "операция“ (action)
- Подсистема защиты определяет, разрешить ли данному субъекту выполнить данную операцию над данным объектом
  - например, вы можете читать и изменять (писать в) свои файлы, другие – не могут
  - например, вы можете читать содержимое `/etc/motd`, но не можете изменять его