Прикладные физико-технические и компьютерные методы исследований

Семинар 9

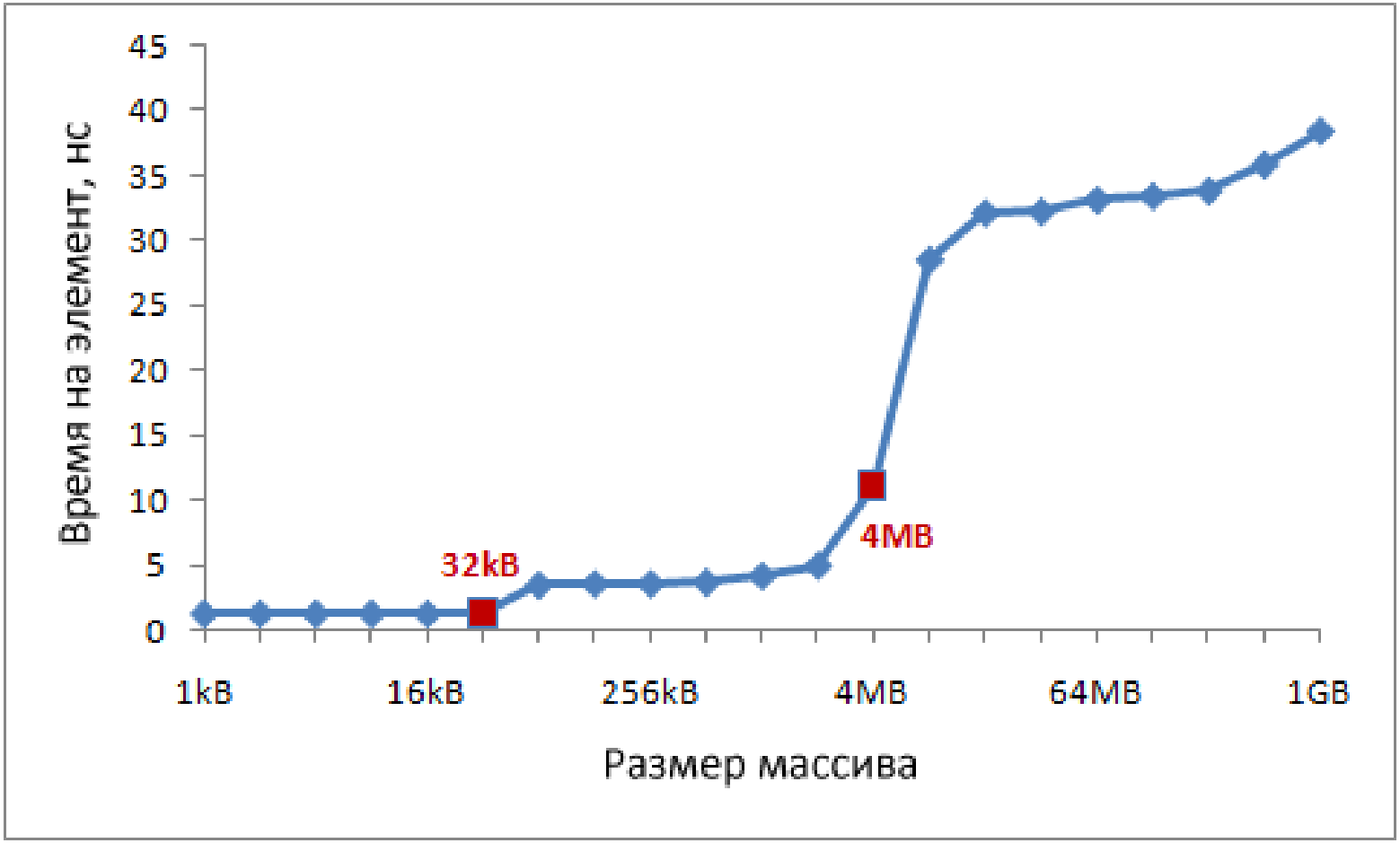
Управление памятью

- Память разбита на ***сегменты.***
- Разделение памяти *на* ***физическую*** *и* ***логическую (виртуальную)****.*
- Идея ***локальности*** (*Выигрыш от многоуровневой памяти*).
- На жёстком диске данные хранятся и без наличия питания. В оперативной памяти, кэшах, регистрах – только пока есть питание.

Идея локальности. Кэширование.

http://habrahabr.ru/post/93263/

Идея локальности. Кэширование.



Виртуальная память

- Адресация пространства, гораздо большего, чем емкость физической памяти.
- Только необходимая на данный момент информация хранится в оперативной памяти.

Виртуальная память

- Как отобразить виртуальную память в физическую?
- Таблица, отображающая память побайтно была бы размером с саму память.
- Выход: разбить виртуальную память на ***страницы*** и соответствующие им ***кадры*** в физической памяти и осуществлять отображение уже не на уровне байт, а на уровне страниц.

Виртуальная память

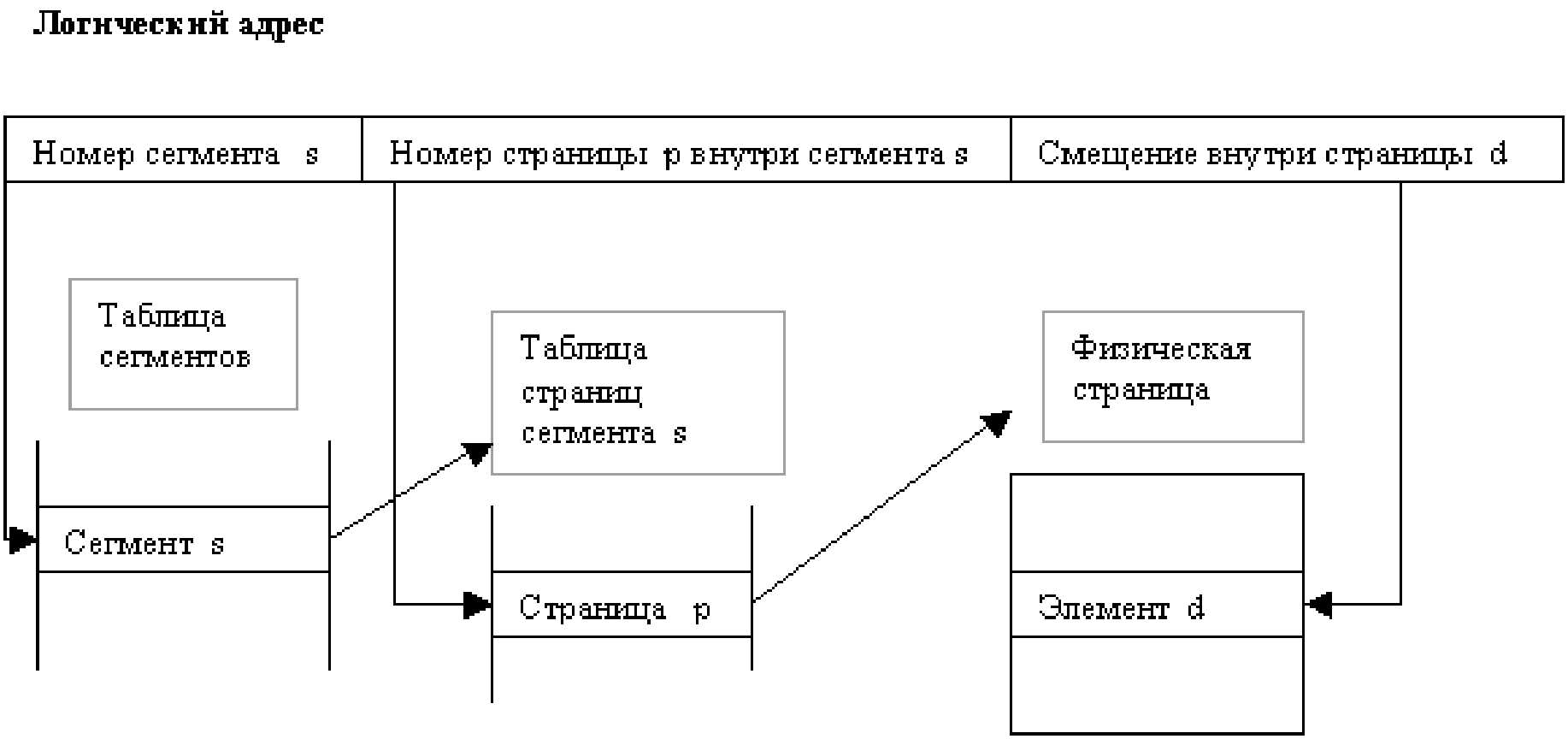
- Контроль доступа к конкретной ячейке памяти *(page, shift)*.
- Для каждого процесса своя таблица страниц.

Виртуальная память

- Память удобно разбивать на сегменты – области
- Память пользователя – набор сегментов (сегмент кода, сегмент данных, стек и т.д.)
- Адрес памяти = номер сегмента, номер страницы, смещение относительно начала страницы.

Виртуальная память

- Сегментно-страничная адресация



Промежуточные выводы

- Разделение памяти *на **физическую** и **логическую (виртуальную)***.
- Идея ***локальности*** (*Выигрыш от многоуровневой памяти*).
- Нельзя подобрать оптимальные сразу для всех задач размеры для количества регистров, размер и количество кэшей, размер страниц и т.д.
- …Скорость обращения к данным, фрагментация данных, объём дополнительной информации

Промежуточные выводы

- Быстродействие зависит от алгоритмов работы файловой системы, т.е. алгоритмов обновления данных в более «быстрой» данными более «медленной».
- Все эти алгоритмы полагаются на свойство ***локальности***, т.е. чем более локальные (предсказуемые) обращения к памяти вы используете, тем быстрее работает программа.
- Чем «проще» вы пишете код, тем больше шанс, что компилятор его хорошо прооптимизирует.
- Чем «предсказуемее» работает программа, тем проще процессору «угадывать» последующие команды, что сказывается положительно на производительности.

Файловая система

- До этого говорили про управление памятью: про сегменты и страницы. Под физической памятью понимали фактически оперативную память (ОЗУ, RAM – Random Access Memory).
- Далее будем говорить про то, как данные в виде файлов расположены на жёстком диске (Винчестер, HDD - Hard Disk Drive)

Файл на диске

- Разбит на блоки по 4К (не обязательно смежных).
- Адреса блоков для данного файла хранятся в виде списка отдельно в т.н. ***индексе файла.***
- Наличие древовидной структуры папок является просто дополнительным индексом, т.о. вся файловая система – большой индексированный файл.

Обращение к жесткому диску

- В $\sim 10^5$ раз медленнее, чем к оперативной памяти.
- Есть целое отдельное направление разработки алгоритмов для работы с данными, которые не влезают в оперативную память.

Типы файлов

- Регулярные файлы (ASCII – текстовая информация, бинарные – произвольная информация)
- Папки
- Если файл слишком большой, то размер индекса тоже большой->увеличивается время обращения к данным. Тогда строят индексы для индексов.

Атрибуты файла

- Хранятся в Index Node
- Содержат
  - Тип файла и права различных категорий пользователей для доступа к нему.
  - Идентификаторы владельца-пользователя и владельца-группы.
  - Размер файла в байтах (только для регулярных файлов, директорий и файлов типа "связь").
  - Время последнего доступа к файлу.
  - Время последней модификации файла.
  - Время последней модификации самого индексного узла.
- Суперблок

Работа с файлами

- open, close
- stat, fstat, lstat – чтение атрибутов файла
- chmod, chown, chgrp – изменение атрибутов файла
- lseek – изменение положения указателя текущей позиции
- opendir, readdir, rewinddir, closedir

Упражнение 1

- С помощью opendir, readdir, closedir и stat выписать список файлов, расположенных внутри текущей директории с указанием их типов (regular file, directory) и размера.

Упражнение 2а (домашнее упражнение)

- Необходимо реализовать поиск файла на диске. Аргументом командной строки задаётся исходная директория, максимальная глубина поиска и имя файла, который ищем, например

./a.out . 2 myfile.txt

- P.S. При открытии директории (opendir) в качестве аргумента нужно передать либо абсолютный путь до неё, либо путь относительно текущей директории, откуда была запущена программа.

Упражнение 2б (если 2а кажется очень сложным)

- Написать программу копирования файла включая атрибуты (stat + chmod, chown, chgrp). Имя исходного файла и копии задаются как аргументы командной строки.