МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА АСЦТ

ОТКРЫТЫЙ УРОК

по дисциплине «Операционные системы»

для магистрантов очного отделения, обучающихся по специальности:
710100 «Информатика и вычислительная техника»

Тема урока: Управление памятью.
Аппаратное управление памятью.

Группа: ИВТ(м)-1-20

Составитель: доцент Кудуев А.Ж.

2020-2021 – учебный год

Цели и задачи урока:

- Ознакомление с основными понятиями организации файловой системы, обслуживание файловой структуры
- Знает основные понятия управление памятью и определяет принцип работы.
- Организовать деятельность учащихся по изучению управление памятью и аппаратное управление памятью.
- формировать практические умения работы с управлением памятью.

Тип урока: урок объяснения нового материала.

Вид урока: теоретический урок с элементами практикума.

Оборудование: интерактивная доска, тетради, карточки, компьютерная презентация.

<u>Формируемые общие компетенции:</u>

- ✓ ИК-4: способен к дальнейшему самостоятельному обучению;
- ✓ ПК-9: способен разрабатывать проекты автоматизированных систем различного назначения, обосновывать выбор аппаратно-программных средств автоматизации и информатизации предприятий и организаций;
- ✓ ПК-13: способен проектировать и применять инструментальные средства реализации программно-аппаратных проектов

План урока:

1. Организационный момент (1 мин.)
2. Теоретическая часть (20 мин.)
3. Практическая часть (25 мин.)
4. СРС (2 мин.)
5. Итог урока (2 мин.)

Ход занятия

1. Организационный момент:

- *Приветствие;*
- *Выявление отсутствующих;*
- *Объявление даты, темы и цели урока;*
- *Ознакомление с ходом и формами работы на уроке;*
- *Мотивационный момент;*
- *Домашние задание (СРС).*

2. Теоретическая часть.

Тема: Управление памятью

Аппаратное управление памятью

Большинство компьютеров используют большое количество различных запоминающих устройств, таких как: ПЗУ, ОЗУ, жесткие диски, магнитные носители и т.д. Все они представляют собой виды памяти, которые доступны через разные интерфейсы. Два основных интерфейса — это прямая адресация процессором и файловые системы. Прямая адресация подразумевает, что адрес ячейки с данными может быть аргументом инструкций процессора.

Режимы работы процессора x86:

- *реальный* — прямой доступ к памяти по физическому адресу
- *защищенный* — использование виртуальной памяти и колец процессора для разграничения доступа к ней

Виртуальная память

Виртуальная память — это подход к управлению памятью компьютером, который скрывает физическую память (в различных формах, таких как: оперативная память, ПЗУ или жесткие диски) за единым интерфейсом, позволяя создавать программы, которые работают с ними как с единым непрерывным массивом памяти с произвольным доступом.

Решаемые задачи:

- поддержка изоляции процессов и защиты памяти путём создания своего собственного виртуального адресного пространства для каждого процесса

- поддержка изоляции области ядра от кода пользовательского режима
- поддержка памяти только для чтения и с запретом на исполнение
- поддержка выгрузки не используемых участков памяти в область подкачки на диске (свопинг)
- поддержка отображённых в память файлов, в том числе загрузочных модулей
- поддержка разделяемой между процессами памяти, в том числе с копированием-при-записи для экономии физических страниц

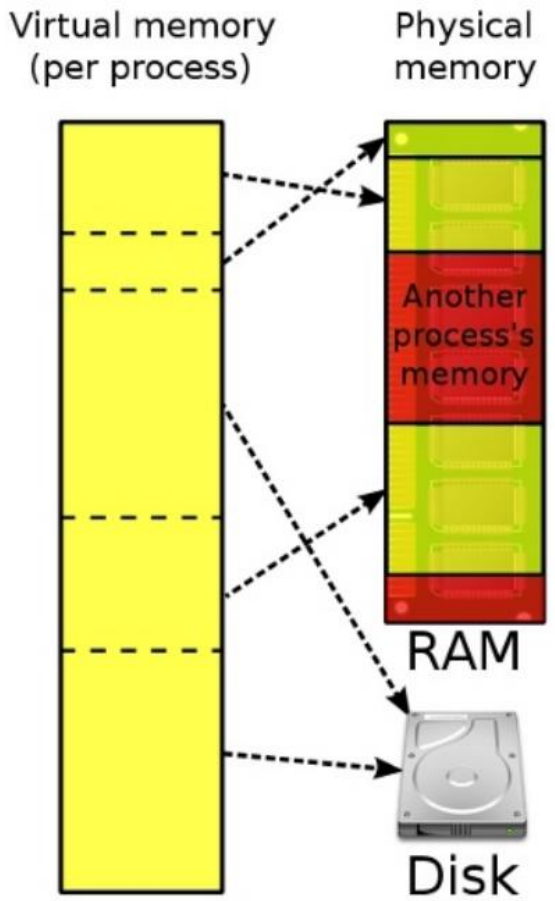

Рис. 1. Абстрактное представление виртуальной памяти

Виды адресов памяти:

- *физический* - адрес аппаратной ячейки памяти
- *логический* - виртуальный адрес, которым оперирует приложение

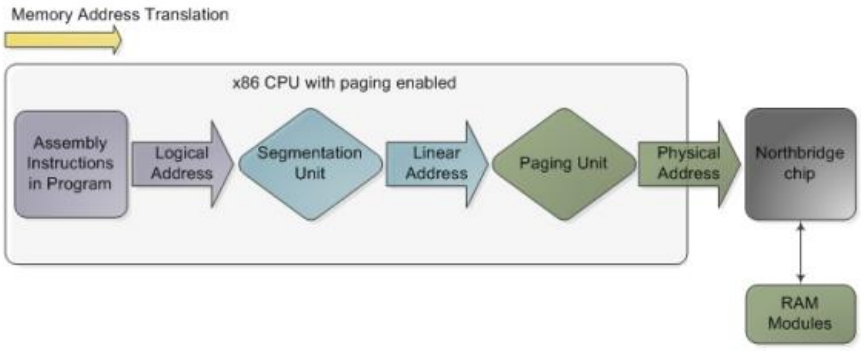

Рис. 2. Трансляция логического адреса в физический

За счет наличия механизма виртуальной памяти компиляторы прикладных програм могут генерировать исполняемый код в рамках упрощенной абстрактной линейной модели памяти, в ко торой вся доступная память представляется в виде непрерывного массива машинных слов, адресуемого с 0 до максимально возможного адреса для данной разрядности (2^N, где N -количество бит, т.е. для 32-разрядной архитектуры максимальный адрес — 2^32 = #FFFFFFFF). Это значит что резуль тирующие программы не привязаны к конкретным параметрам запоминающих устройств, таких как их объем, режим адресации и т.д.

Кроме того, этот дополнительный уровень позволяет через тот же самый интерфейс обращения к данным по адресу в памяти реализовать другие функции, такие как обращение к данным в файле (через механизм mmap) и т.д.

Наконец, он позволяет обеспечить более гибкое, эффективное и безопасное управление памятью компьютера, чем при использовании физической памяти напрямую.

На аппаратном уровне виртуальная память, как правило, поддерживается специальным устройством - Модулем управления памятью.

Страничная организация памяти

Страничная память — способ организации виртуальной памяти, при котором единицей отображения виртуальных адресов на физические является регион постоянного размера — страница.

Рис. 3. Трансляция адреса в страничной модели

При использовании страничной модели вся виртуальная память делится на N страниц таким образом, что часть виртуального адреса интерпретируется как номер страницы, а часть — как смещение внутри страницы. Вся физическая память также разделяется на б локи такого же размера — **фреймы**. Таким образом в один фрейм может быть загружена одна страница. **Свопинг** — это выгрузка страницы из памяти на диск (или друг ой носитель большего объема), который используется тогда, когда все фреймы заняты. При этом под свопинг попадают страницы памяти неактивных на данный момент процессов.

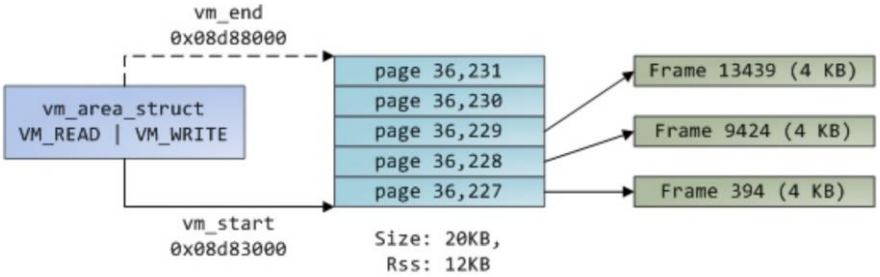

Рис. 4. Память процесса в страничной модели

Таблица соответствия фреймов и страниц называется таблицей страниц. Она одна для всей системы. Запись в таблице страниц включает служебную информацию, такую как: индикаторы доступа только на чтение или на чтение/запись, находится ли страница в памяти, производилась ли в нее запись и т.д. Страница может находится в трех состояниях: загружена в память, выгружена в своп, еще не загружена в память (при изначальном выделении страницы она не всегда сразу размещается в памяти).

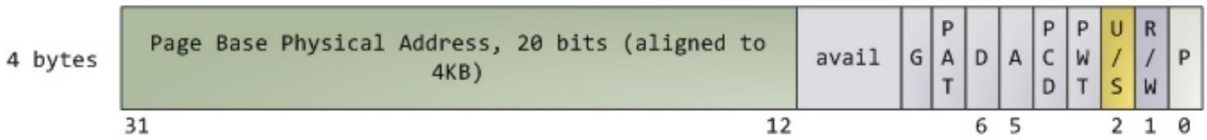

Рис. 5. Запись в таблице страниц

Размер страницы и количество страниц зависит от того, какая часть адреса выделяется на номер страницы, а какая на смещение. К примеру, если в 32-разрядной системе разбить адрес на две равные половины, то количество страниц будет составлять 2^16, т .е. 65536, и размер страницы в байтах будет таким же, т.е. 64 КБ. Если уменьшить количество страниц до 2^12, то в системе будет 4096 страницы по 1МБ, а если увеличить до 2^20, то 1 миллион страниц по 4КБ. Чем больше в системе страниц, тем больше занимает в памяти таблица страниц, соответственно работа процессора с ней замедляется. А поскольку каждое обращение к памяти требует обращения к таблице страниц для трансляции виртуального адреса, такое замедление очень нежелательно. С другой стороны, чем меньше страниц и, соотвественно, чем они больше по объему — тем больше потери памяти, вызванные внутренней фрагментацией страниц, поскольку страница является е диницей выделения памяти. В этом заключается диллема оптимизации страничной памяти. Она особенно актуальна при переходе к 64-разрядным архитектурам.

Для оптимизации страничной памяти используются следующие подходы:

- специальный кеш — TLB (translation lookaside buffer) — в котором хранится очень небольшое число (порядка 64) наиболее часто используемых адресов страниц (основные страницы, к которым постоянно обращается ОС)

- многоуровневая (2, 3 уровня) таблица страниц — в этом случае виртуальный адрес разбивается не на 2, а на 3 (4,...) части. Последняя часть остается смещением внутри страницы, а каждая из остальных задает номер страницы в таблице страниц 1-го, 2-го и т.д. уровней. В этой схеме для трансляции адресов нужно выполнить не 1 обращение к таблице страниц, а 2 и более. С другой стороны, это позволяет с вопить таблицы страниц 2-го и т.д. уровней, и подгружать в память только те таблицы, которые нужны текущему процессу в текущий момент времени или же даже кешировать их. А каждая из таблиц отдельного уровня имеет существенно меньший размер, чем имела бы одна таблица, если бы уровень был один

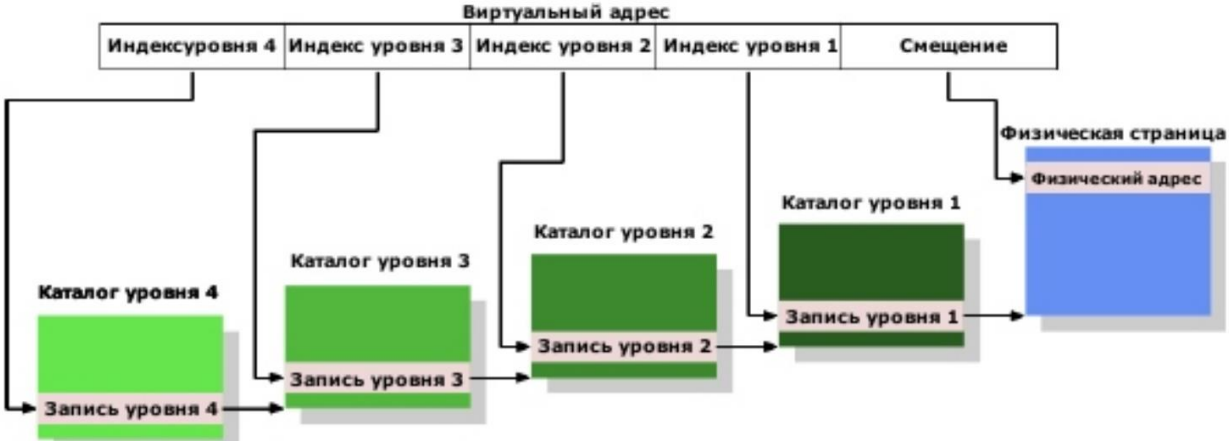

Рис. 6. Многоуровневная система страниц

➢ инвертированная таблица страниц — в ней столько записей, сколько в системе фреймов, а не страниц, и индексом является номер фрейма: а число фреймов в 64- и более разрядных архитектурах существенно меньше теоретически возможного числа страниц. Проблема такого подхода — долгий поиск виртуального адреса. Она решается с помощью таких механизмов как: хеш-таб лицы или кластерные таблицы страниц

Сегментная организация памяти

Сегментная организация виртуальной памяти реализует следующий механизм: вся память делиться на сегменты фиксированной или произвольной длины, каждый из которых характеризуется своим начальным адресом — базой или селектором. Виртуальный адрес в такой системе состоит из 2-х компонент: базы сегмента, к которому мы хотим обратиться, и смещения внутри сегмента. Физический адрес вычисляется по формуле:

addr = base + offset

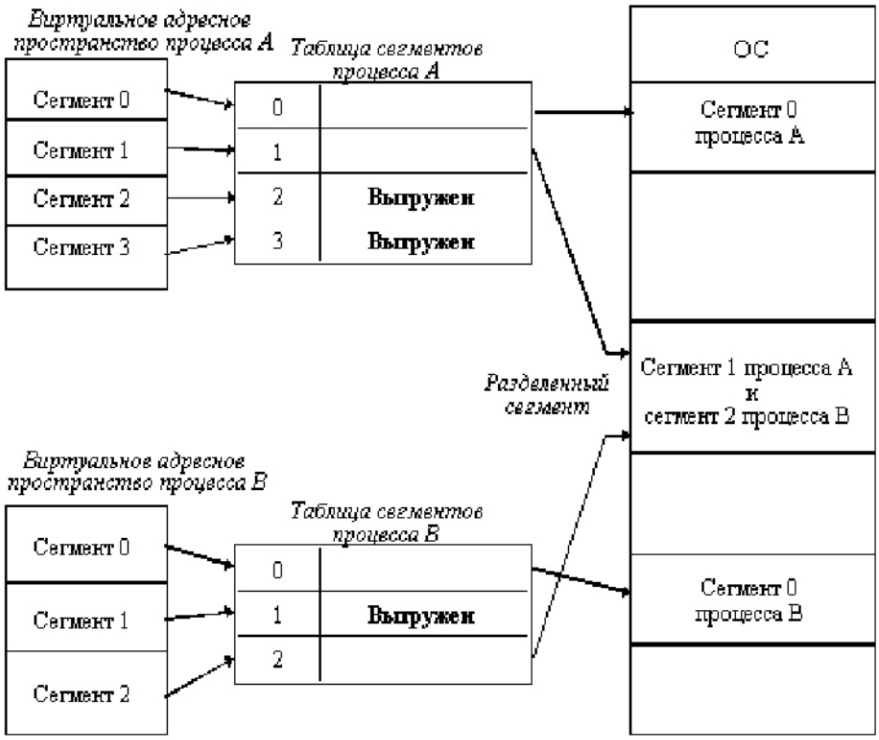

Рис. 7. Представление сегментной модели виртуальной памяти

Историческая модель сегментации в архитектуре x86

В архитектуре x86 сегментная моде ль памяти была впервые реализована на 16-разрядных процессорах 8086. Используя только 16 разрядов для адреса давало возможность адресовать только 2^16 байт, т.е. 64КБ памяти. В то же время стандартный размер физической памяти для э тих процессоров был 1МБ. Для того, чтобы иметь возможность работать со всем доступным объемом памяти и была использована сегментная модель. В ней у процессора было выделено 4 специализированных регистра CS (сегмент кода), SS (сегмент стека), DS (сегмент данных), ES (расширенный сегмент) для хранения базы текущего сегмента (для кода, стека и данных программы).

addr = base << 4 + offset

Это приводило к возможности адресовать большие адреса, чем 1МБ — т.н. Gate A20.

См. также: http://en.wikipedia.org/wiki/X86_memory_segmentation

Плоская модель сегментации

32-разрядный процессор 80386 мог адресовать 2^32 байт памяти, т.е. 4ГБ, что более чем перекрывало доступные на тот момент размеры физической памяти, поэтому изначальная причина для использования сегментной организации памяти отпала.

Однако, помимо особого способа адресации сегментная модель также предоставляет механизм защиты памяти через кольца безопасности процессора: для каждого сегмента в таблице сегментов задается значение допустимого уровня привилегий (DPL), а при обращении к сегменту передается уровень привилегий текущей программы (запрошенный уровень привилегий, RPL) и, если RPL > DPL доступ к памяти запрещен. Таким образом обеспечивается защита сегментов памяти ядра ОС, которые имеют DPL = 0. Также в таблице сегментов задаются другие атрибуты сегментов, такие как возможность записи в память, возможность исполнения кода из нее и т.д.

Таблица сегментов каждого процесса находится в памяти, а ее начальный адрес загружается в регистр LDTR процессора. В регистре GDTR процессора хранится указатель на глобальную таблицу сегментов.

В современных процессорах x86 используется "Плоская модель сегментации", в которой база всех сегментов выставлена в нулевой адрес.

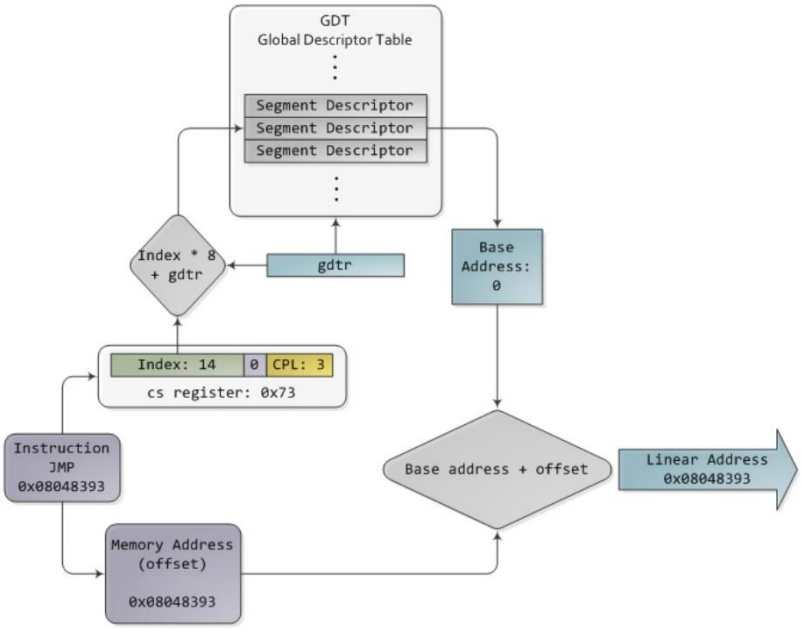

Рис. 8. Плоская модель сегментации

Виртуальная память в архитектуре x86

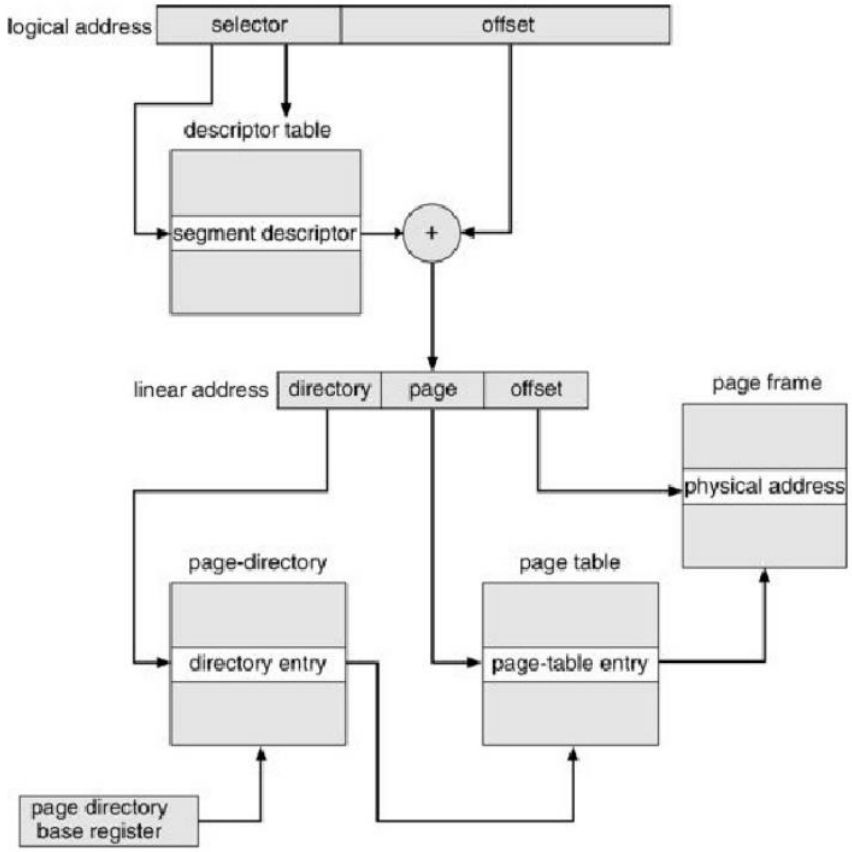

Рис. 9. Трансляция адреса в архитектуре x86

Системные вызовы для взаимодействия с подсистемой виртуальной памяти:

- brk, sbrk - для увеличения сегмента памяти, выделенного для данных программы

- mmap, mremap, munmap - для отображения файла или устройства в память mprotect - изменение прав доступа к областям памяти процесса

Пример выделение памяти процессу:

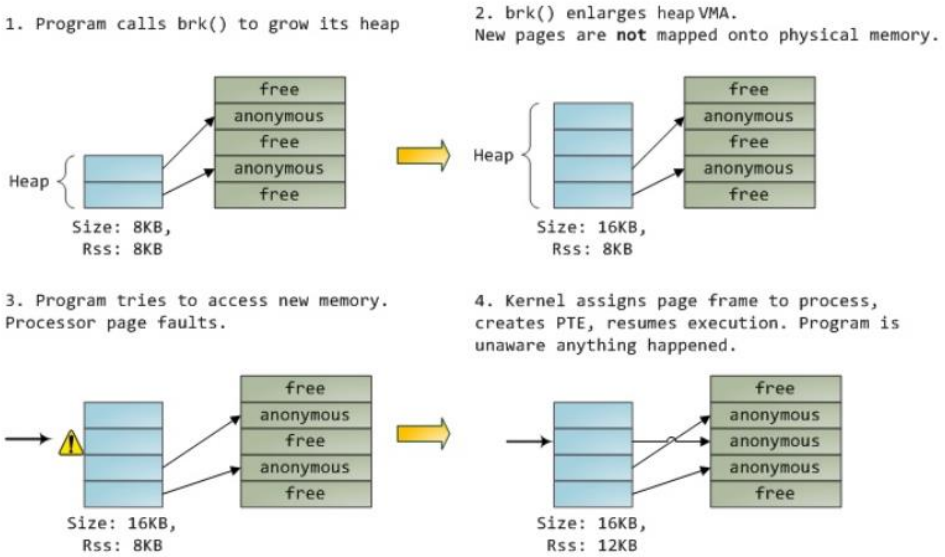

Рис. 10. Ленивое выделение памяти при вызове brk

Домашние задание (СРС)

1. Алгоритмы выделения памяти
2. Кеширование
3. Алгоритмы замещения записей в кеше

Учебная литература:

1. Дейтел Г. Введение в операционные системы: В 2 т.: Пер. с англ. М.: Мир, 1987.
2. Проскурин В.Г., и др. Программно-аппаратные средства обеспечения информационной безопасности. Защита в операционных системах: Уч. пособие для вузов / М.:Радио и связь, 2000. 168 с.
3. Хелен Кастер. Основы Windows NT и NTFS: Пер. с англ. М.: Издательский отдел «Русская редакция» ТОО «ChannelTradingLtd.», 1996. 440 с.
4. Пол Мак-Федрис Microsoft Windows Vista. Полное руководство = Microsoft Windows Vista Unleashed. — М.: "Вильямс", 2007. — С. 864. — ISBN 978-5-8459-1302-9
5. Брайан Ливингстон, Пол Таррот Секреты Microsoft Windows Vista = Windows Vista Secrets. — М.: "Диалектика", 2007. — С. 456. — ISBN 0-7645-7704-2
6. Торвальдс, Л.; Даймонд, Д. Ради удовольствия = Just for fun. — М.: ЭКСМО-Пресс, 2002. — С. 288. — ISBN 5-04-009285-7
7. Роберт Лав Разработка ядра Linux = Linux Kernel Development. — 2-е изд. — М.: "Вильямс", 2006. — С. 448. — ISBN 0-672-32720-1
8. Родригес К. З., Фишер Г., Смолски С. Linux: азбука ядра.. — "КУДИЦ-ПРЕСС", 2007. — С. 584. — ISBN 978-5-91136-017-7
9. Баррет Д. Linux: основные команды. Карманный справочник. 2-е издание.. — "КУДИЦ-ПРЕСС", 2007. — С. 288. — ISBN 5-9579-0050-8