Билет 1

1. ОС — определение ОС. Ресурсы вычислительной системы. Режимы ядра и задачи: переключение в режим ядра — классификация событий. Процесс, как единица декомпозиции системы, диаграмма состояний процесса с демонстрацией действий, выполняемых в режиме ядра. Переключение контекста. Потоки: типы потоков, особенности каждого типа потоков.

ОС

ОС (для обычного пользователя) - это интерфейс.

ОС (для профессионала) - это менеджер ресурсов вычислительных машин.

ОС- комплект программ, которые совместно управляют ресурсами вычислительной системы и процессами, использующими эти ресурсы при вычислениях

Классификация ОС

1. *Однопрограммные пакетной обработки*
2. *Мультипрограммная пакетной обработки*
3. *Мультипрограммная с разделением времени В оперативной памяти нах. большое число программ, процессорное время квантуется, чтобы обеспечить гарантированное время ответа системы. Время ответа системы не должно превышать 3 секунд.*
4. *Системы реального времени*
5. *серверные ОС – предоставляющие доступ к аппаратным (принтеры) и программным ресурсам (файлы доступа к Интернет) из сети.*
6. *многопроцессорные*
7. *встроенные ОС (телевизоры, микроволновые печи).*
8. *опер системы для смарт-карт (самые маленькие операционные системы).*

Ресурсы вычислительной системы

Ресурс - любой из компонентов вычислительной системы и предоставляемые ею возможности (как аппаратные, так и программные)

Основная задача ОС – управление процессами и выделение процессам ресурсов.

Ресурсы вычислительной системы (первые 2 – главные)

1. Процессорное время
2. Объем памяти (оперативной). (Объем физической памяти (ОЗУ))
3. Внешние устройства (Устройства ввода/вывода (УВВ))
4. Ключи защиты
5. Реентерабельные коды самой системы (код чистой процедуры-процедуры, не модифицирующей саму себя, то есть данные (переменные). Из реентерабельных кодов вынесены данные.)
6. Данные. Данные в ОС находятся в специальных системных «таблицах» (условно).
7. *Каналы*
8. *Таймер*
9. *Семафоры*
10. *Разделяемая память*

Режимы ядра и задачи:

С точки зрения UNIX процесс часть времени выполняется в режиме ядра (выполняется реентерабельный код ОС, а часть – в режиме пользователя (режим задачи) (выполняется собственный код). Таким образом, процесс постоянно переключается между режимами.

Режим ядра - привилегированный режим (в нем выполняется реентерабельный код ОС). Те части NT, которые исполняются в режиме ядра, такие как драйверы устройств и подсистемы типа Диспетчера Виртуальной Памяти, имеют прямой доступ ко всей аппаратуре и памяти. Различия в работе программ пользовательского режима и режима ядра поддерживаются аппаратными средствами компьютера (а именно - процессором).

Пользовательский режим (режим задачи) - наименее привилегированный режим, он не имеет прямого доступа к оборудованию и у него ограниченный доступ к памяти.

Переключение в режим ядра - классификация событий

3 события, переводящие систему в режим ядра:

1. Системные вызовы

Можно вызвать из программы с помощью команды int. Часто называются программными прерываниями (software interrupts (trups)). СВ – синхронные (по отношению к выполняемой программе) события, которые происходят в процессе работы программы.

СВ-тот интерфейс, который предоставляет пользователю ОС (Application function interface API) - набор функций, определенных в системе, которые может использовать приложение, чтобы получать сервис (обслуживание) системой. Например, ввод/вывод

ОС старается минимизировать количество СВ особенно ввод/вывод, так как они связаны с обращением к внешним устройствам, а в UNIX все файл, даже устройства, чтобы со всеми устройствами система работала единообразно (read/write)

Почему обращение к внешним устройствам – системный вызов? ОС не позволяет программам напрямую обращаться, так как иначе был бы открыт доступ к структуре ядра.

2. Исключения (исключительные ситуации). Делятся на исправимые (страничное прерывание) и неисправимые (ошибки (/0) ошибка деления, ошибки адресации). Являются синхронными событиями по отношению к коду

3. Аппаратные прерывания (interrupts) - прерывания, поступившие от устройств (от таймера, от УВВ (устройства ввода вывода) и т.д.). (поступают от контроллера прерываний) асинхронные события в системе происходят вне зависимости от какой-либо работы, выполняемой процессором

Всегда отдельно рассматривается прерывание от системного таймера – особое и единственное периодическое прерывание с важнейшими системными функциями.

В системах разделения времени – декремент кванта

Прерывание от внешнего устройства по завершении операции ввода/вывода – внешние устройства информируют процессор о том, что ввод/вывод завершен и процесс может перейти к обработке. При этом (даже вывод) происходит получение данных об успешности или неуспешности завершения операции.

Отдельно – прерывание от действий оператора (win: ctrl+alt+delete, unix :ctrl+C)

Процесс, как единица декомпозиции системы.

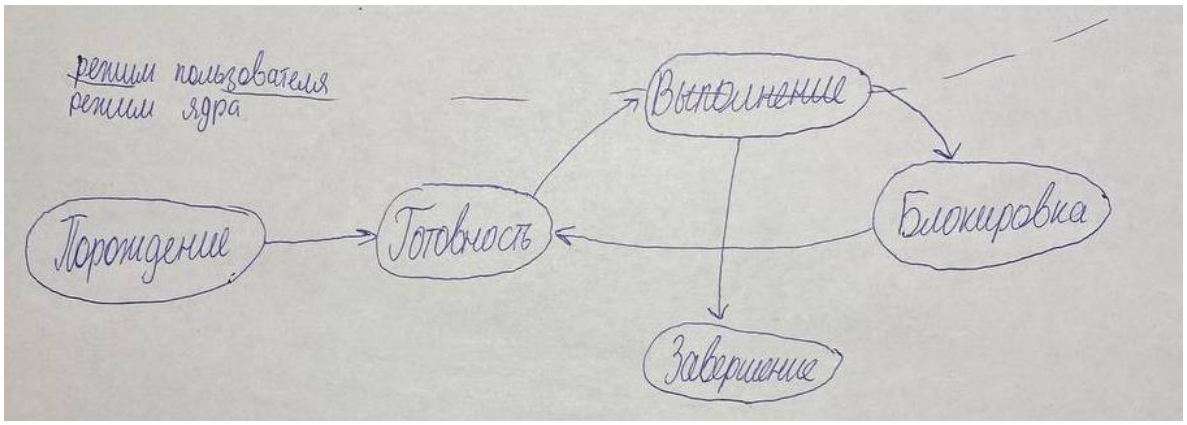
Процесс - программа в стадии выполнения.

В любой ОС основной абстракцией является процесс.

диаграмма состояний процесса с демонстрацией действий, выполняемых в режиме ядра.

Или этапы жизни процесса

(только состояния, которые есть в любой ОС, разработчики сами для себя выбирают дополнительные состояния.)



ОС – тоже программа и работает по тем же принципам. Для обработки данных нужно объявить переменную. Так и процесс должен быть определен (иметь имя)

Порождение - присвоение процессу строки в таблице процессов.

(UNIX:)

- Первый шаг системы при запуске программы – присвоение процессу идентификатора. Но надо управлять процессом (выделять ресурсы), следовательно, должна существовать структура, которая описывает процесс.
- Выделяется строка в таблице процессов: номер=идентификатор (целое число), а таблица-массив структур.
- Инициализация структуры, описывающей процесс (тех полей, которые можно)
- Выделить 1 ресурс – память.

После того, как процесс получил все необходимые ресурсы (кроме времени) – состояние готовности

В многозадачных ОС (операционных многозадачных системах пакетной обработки и системах разделения времени) в состоянии готовность одновременно может быть большое количество процессов, они организуют очередь, а в однозадачных (DOS, однозадачной пакетной обработки) – только один.

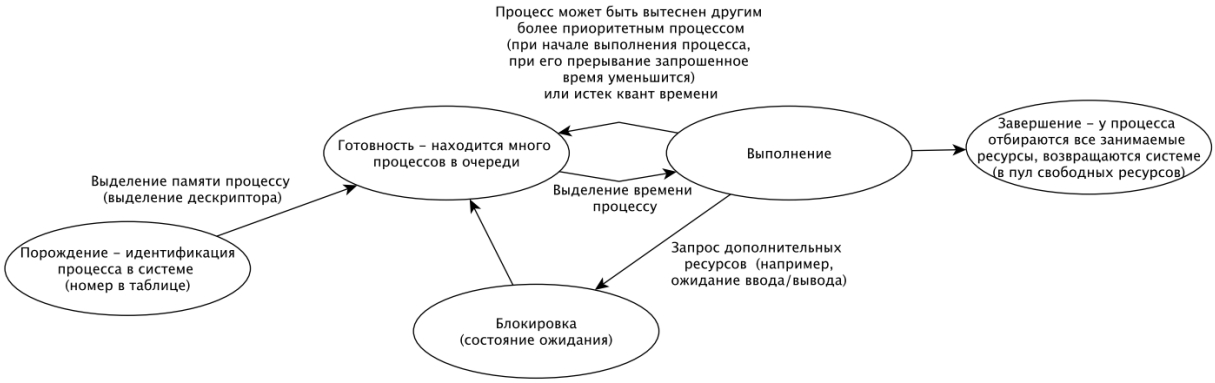
После того, как процесс в стадии готовности получил последний ресурс-время-он переходит в состояние выполнения. Из состояния выполнения он может перейти:

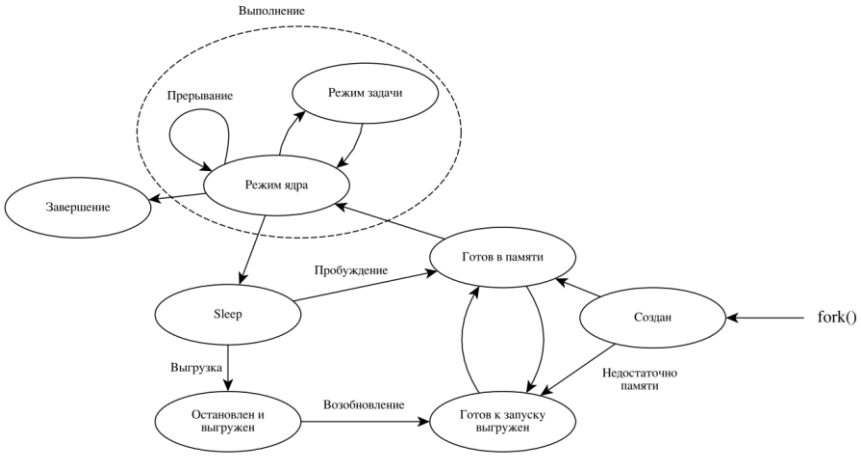
- в состояние блокировки, если он запросил ресурс, который не может быть выделен моментально, или запросил ввод/вывод. Получив необходимый ресурс, процесс перейдет в состояние готовности.
- в состояние завершения. Забираются все его ресурсы и возвращаются системе.

Вот именно в состоянии выполнения часть времени в режиме ядра, часть – в режиме пользователя.

При многозадачности процесс может вытиснятся, если пришел более приоритетный процесс.

Вытеснение может произойти, если истек квант





Переключение контекста

Переключение контекста - текущий процесс уступает процессорное время другому процессу

(Вахалия, 68 с.)

Переключение контекста - сохранение аппаратного контекста текущего процесса в области u (Блок Управления Процессом (БУП)) и загрузка аппаратного контекста другого процесса из БУП. Переключение процессора с одного процесса на другой.

В системах квантования при истечении кванта, процесс возвращается в очередь. При вытеснении происходит переключение контекста. Если процесс вытеснен или исчерпан

квант, то необходимо сохранить информацию, которая поможет вернуться к продолжению выполнения.

Полный контекст - содержит информацию о выделенных процессу ресурсов (отражено в дескрипторе процесса), сохранение в оперативной памяти.

Аппаратный контекст - регистры общего назначение + IP + SP + PSW + регистры управления памятью и сопроцессора

Аппаратный контекст $\in$ Полный контекст.

Потоки: типы потоков, особенности каждого типа потоков.

Поток - это непрерывная часть кода процесса, выполняющаяся параллельно с другими частями кода программы.

Поток не имеет своего адресного пространства, а выполняется в адресном пространстве процесса.

Типы потоков.

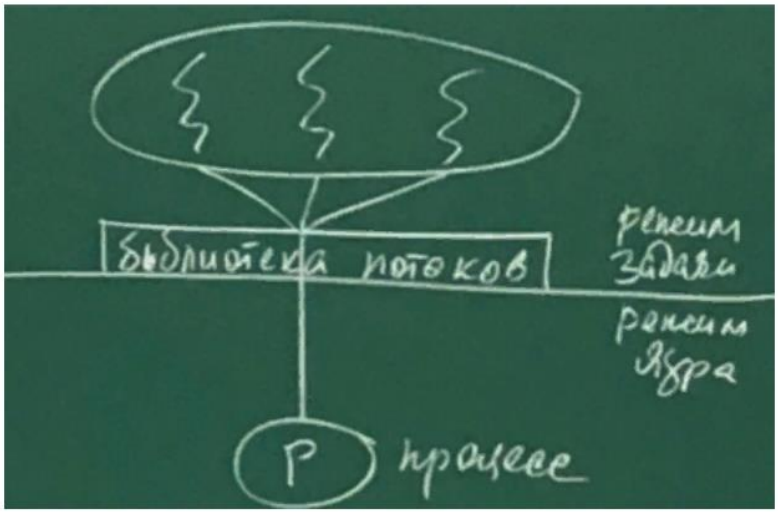
1. Режим пользователя.

Пользовательские потоки (threads) - это потоки о которых ядру ничего неизвестно.

Чтобы обратиться к вводу/выводу блокируются все процессы и происходит переход в режим процессора.

Для управления необходима специальная библиотека потоков, она предоставляет функционал для работы с потоками.

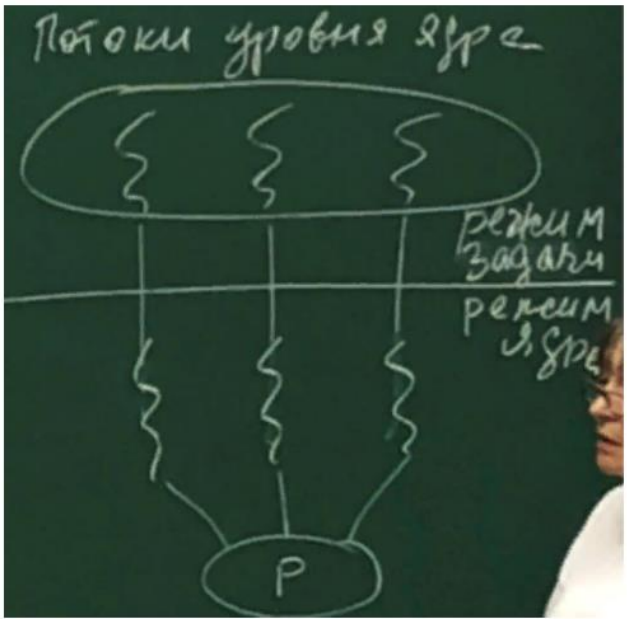
<!--(Она должна предоставлять функции создания и удаления, планирования потоков, диспетчеризации, сохранения контекста, обеспечивать возможность взаимодействия потоков. Как только поток запросил функции ядра, то процесс приостанавливается, пока не завершится операция ядра (например IO))-->



2. Режим ядра.

Владельцем ресурсов является процесс.

Если в программе никакие потоки не создаются, то для общности создается один главный поток.



2. Три режима компьютера на базе процессоров Intel (X86). Адресация аппаратных прерываний в защищенном режиме: таблица дескрипторов прерываний (IDT) — формат дескриптора прерываний, типы шлюзов. Пример заполнения IDT из лабораторной работы.

Три режима

Компьютеры на базе процессоров Intel работают в 3 режимах:

1. Реальный – 16р режим с 16-р регистрами и с 20р шиной адреса, intel 8086. 2^20=позволяют адресовать до 1024Кбайт=1 Мбай

Работали под управлением DOS (disk operating system), то есть с внешней дисковой памятью. DOS – однозадачная ОС – в оперативной памяти только одна программа, которая выполняется от начала до конца.

Компьютер начинает работать в реальном режиме (чтобы выполнять меньше команд при загрузке)

В этом режиме используется сегментация памяти (адресация по типу сегмент/смещение, при этом максимальный размер сегмента - 64КБ.)

- Минимальная адресная единица памяти – байт.

2. Защищенный (protected) – 32р режим с 32р регистрами и 32р шиной адреса (4 ГБ).

- В защищенном режиме 4 уровня защиты (4 кольца привилегий). Ядро ОС находится на 0-м уровне. Пользовательские приложения находятся на 3-ем уровне

- Поддерживали 2 независимые схемы управления виртуальной (те фактически несуществующей) памятью – сегментами по запросу и страницами по запросу. Существует аппаратная схема управления памятью. (третья) сегменты, поделенные на страницы – взяли лучшее из 2 схем. Но поддерживаются только первые 2 и они независимые.
- Почему «защищенный»: Windows – система разделения времени. Адресное пространство каждого процесса должно быть защищено, как и адресное пространство ОС.

Существует специальный режим защищенного режима – v86 (virtual). Как задачи в режиме v86 запускаются ОС реального режима. Запускается виртуальная 86 машина, и в этой среде может выполняться одна программа реального режима. (таких виртуальных машин может быть несколько)

Многозадачный. Каждая запущенная виртуальная машина является v86 со всеми вытекающими (1 задача, 1Мб, 16р операнды).

Задача исполняется с самыми низкими привилегиями в кольце 3. Когда в такой задаче возникает прерывание или исключение, процессор переключается из виртуального режима в защищённый. Поэтому все прерывания отображаются в операционную систему, работающую в защищённом режиме. Прерывания обрабатываются обычными обработчиками ОС защищенного режима.

флаг VM регистра EFLAGS равен единице. В этом режиме происходит дизассемблирование.

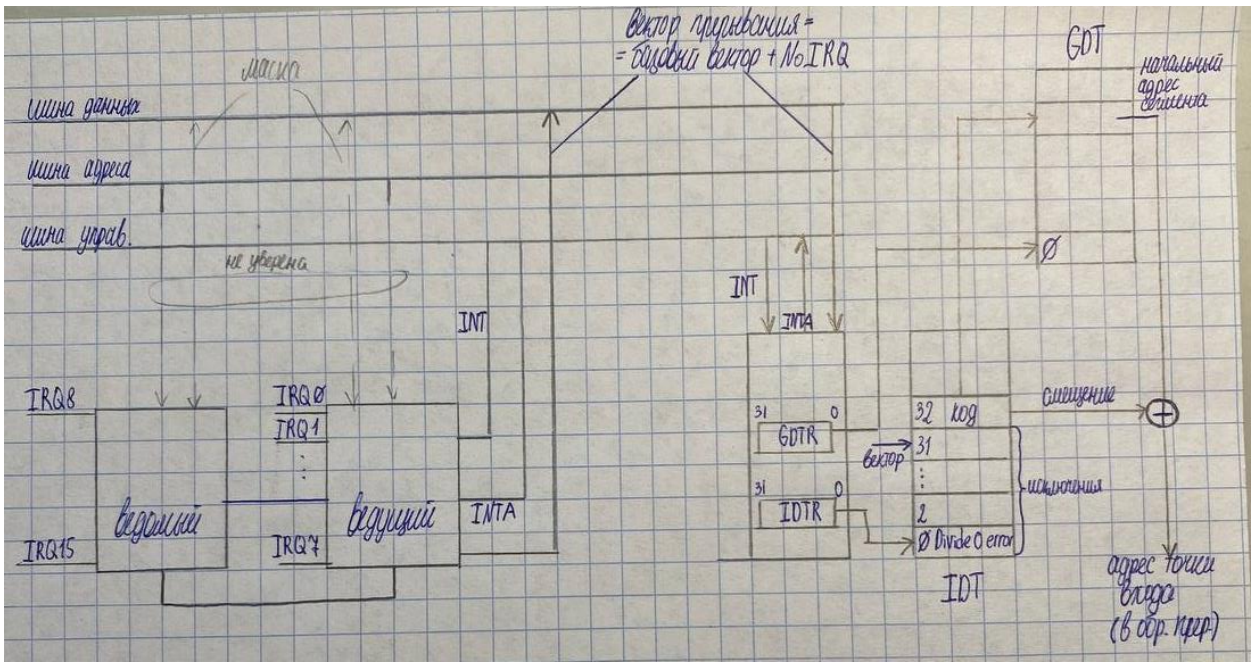
Процессор фактически продолжает использовать схему преобразования адресов памяти и средства мультизадачности защищённого режима.

В виртуальном режиме используется трансляция страниц памяти (в эту область могут быть отображены произвольные страницы памяти). Это позволяет в мультизадачной операционной системе создавать несколько задач, работающих в виртуальном режиме.

3. Long (длинный) – 64р регистры, операнды. Многопроцессность. Только страничная виртуальная память. Поддерживает compatibility – режим совместимости, в котором могут выполняться 32р. Как работает обратная совместимость – рассмотреть регистры. Не поддерживает V86.

но адреса меньше 64-х разрядов (это связано с аппаратными ограничениями).

Адресация аппаратных прерываний в з-р.



(точка входа)

IDTR-32 разрядный регистр, который содержит начальный линейный адрес IDT (все эти регистры есть в каждом ядре).

Контроллер прерывания получает сигнал о прерывании и формирует вектор прерывания, который содержит селектор к IDT.

Взяв значение из регистра IDTR значение базового адреса IDT. В нём по селектору находим дескриптор который уже и содержит селектор, смещение и атрибуты

По селектору выбираем дескриптор из GDT, берём оттуда базовый адрес сегмента и прибавляем его к смещению. Получаем линейный адрес точки входа

или

На вход контроллера приходит сигнал, контроллер формирует сигнал int, который по шине управления переходит на ножку процессора. Процессор посылает intA по шине управления в контроллер. Контроллер выставляет на шину данных вектор прерывания, который поступает в процессор и используется им для адресации обработчика прерывания.

В ЗР для адресации прерывания имеется специальная таблица IDT. Если первые 32 исключения, и мы возьмем 8, то попадем на double fault. Чтобы адресовать прерывания, надо перепрограммировать контроллер на: ведущий-на базовый вектор 32, и этот номер использовать для обращения к таблице. Базового адреса нет, есть смещение и селектор. Отдельно адресуются ведущий и ведомый. От контроллера они могут получить маску??

Схема (Если прерывание не замаскировано):

В РР процессор использует вектор и таблицу векторов прерываний. У ведущего контроллера базовый вектор=8 (8+0=8h) и номер используется для получения смещения к адресу в таблице векторов прерываний. В DOS адрес наз. вектор (4 байт)

В з-р для каждого сегмента программы должен быть определен дескриптор - 8-байтовое поле, в котором в определенном формате записываются базовый адрес сегмента, его длина и некоторые другие характеристики. В р-р сегменты определяются их линейными адресами. (Р-Ф).

Таблица дескрипторов прерываний (IDT)

IDT (interrupt descriptor table) - системная таблица, предназначенная для хранения адресов обработчиков прерываний. (нужна для адресации обработчиков прерываний)

Первые 32 элемента таблицы - под исключения (синхронные события в процессе работы программы) (внутренние прерывания процессора) (в 386 всего 19 исключений (0-19), остальные (20-31) зарезервированы, а *в 486 – и того меньше (реально-18, остальные-зарезервированы)*

32-255 – определяются пользователем (user defined)

Смещение=номер исключения*8

Нам надо адресовать 2 обработчика – таймера и клавиатуры (аппаратные прерывания) через программирование контроллера прерываний. Сейчас прерывания как MSI (message signal interrupts)

- 0-divide error (ошибка деления на 0)
- 8-double fault (если выполнить исключение или маскируемое/немаскрируемое прерывание и возникла ошибка (паника…), завершается работа компьютера)
- 11-segment not present (сегмент отсутствует – надо выполнить определенные действия, чтобы сделать сегмент доступным. Касается управления памятью (нашей программе-не очень))
- 13-general protection (общая защита, должно быть обработано специальным образом. На все исключения-заглушки (double fault-не искл.), а на 13-специальная заглушка (у РФ отражено в структуре таблицы дескрипторов прерываний-без dup)) (нарушение общей защиты (нарушение, код ошибки-та команда), происходит: за пределами сегмента, запрет чтения, за гр. таблицы дескр., int с отс. Номером)
- 14-page fault (fault переводится как исключение, но по-русски здесь прерывание) (страничное прерывание) – обращение к команде/данным, отсутствующим в программе – система должна загрузить нужную страницу. В CR2 адрес, на котором произошло прерывание)

Вектор	Название исключения	Класс исключения	Код ошибки	Команды, вызывающие исключение
0	Ошибка деления	Нарушение	Нет	div, idiv
1	Исключение отладки	Нарушение /ловушка	Нет	Любая команда
2	Немаскируемое прерывание			
3	int 3	Ловушка	Нет	int 3
4	Переполнение	Ловушка	Нет	into
5	Нарушение границы массива	Нарушение	Нет	bound
6	Недопустимый код команды	Нарушение	Нет	Любая команда
7	Сопроцессор недоступен	Нарушение	Нет	esc, wait
8	Двойное нарушение	Авария	Да	Любая команда
9	Выход сопроцессора из сегмента(80386)	Авария	Нет	Команда сопроцессора с обращением к памяти
10	Недопустимый сегмент состояния задачи TSS	Нарушение	Да	jmp, call, iret, прерывание
11	Отсутствие сегмента	Нарушение	Да	Команда загрузки сегментного регистра
12	Ошибка обращения к стеку	Нарушение	Да	Команда обращения к стеку
13	Общая защита	Нарушение	Да	Команда обращения к памяти
14	Страничное нарушение	Нарушение	Да	Команда обращения к памяти
15	Зарезервировано			
16	Ошибка сопроцессора	Нарушение	Нет	esc, wait
17	Ошибка выравнивания	Нарушение	Да	Команда обращения к памяти
18...31 Зарезервированы				
32...255 Предоставлены пользователю для аппаратных прерываний и команд int				

Типы шлюзов.

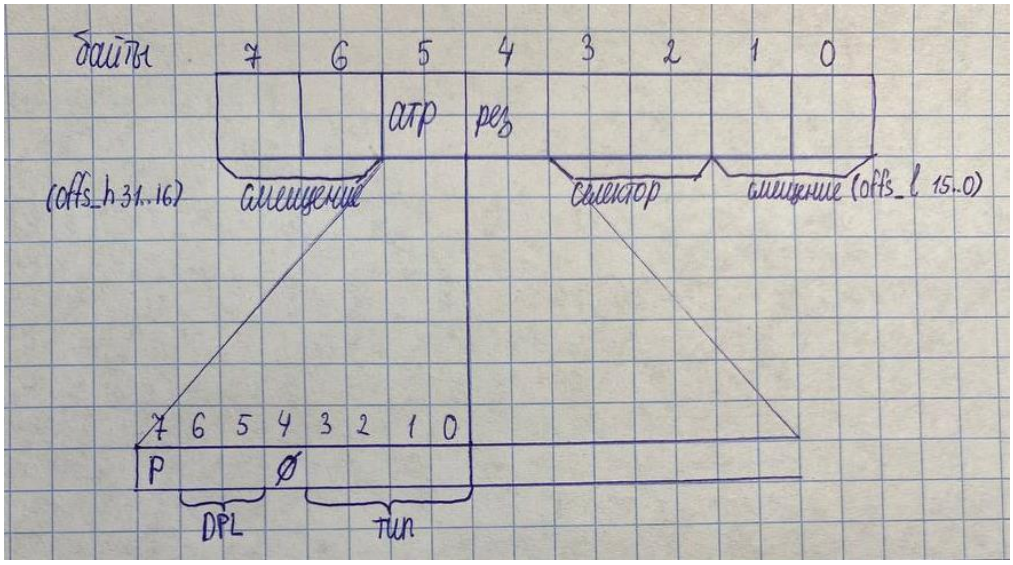
1. Ловушки (trap gate) - обработка исключений и программных прерываний (системных вызовов).

2. Прерывания (interrupt gate) - обработка аппаратных прерываний.

3. Задач (task gate)- *переключение задач в многозадачном режиме.*

Формат дескриптора прерывания

Формат дескриптора (шлюза) для IDT (в скобках – из учебника)



- Байты 0-1 (offs_1), 6-7 (offs_h): 32-битное смещение обработчика

- Байты 2-3 (sel): селектор (сегмента команд) (итого полный 3-хсловный адрес обработчика селектор: смещение)
- Байт 4 зарезервирован
- Байт 5: байт атрибутов - как в дескрипторах памяти за исключением типа:
  Типы: назначение:
  - 0-не определен
  - 1-свободный сегмент состояния задачи TSS 80286
  - 2-LDT
  - 3-занятый сегмент состояния задачи TSS 80286
  - 4-шлюз вызова Call Gate 80286
  - 5-шлюз задачи Task Gate
  - 6-шлюз прерываний Interrupt Gate 80286
  - 7-шлюз ловушки Trap Gate 80286
  - 8-не определен
  - 9- свободный сегмент состояния задачи TSS 80386+
  - Ah-не определен
  - Bh- занятый сегмент состояния задачи TSS 80386+
  - Ch-шлюз вызова Call Gate 80386+
  - Dh- не определен
  - Eh-шлюз прерываний Interrupt Gate 80386+
  - Fh- шлюз ловушки Trap Gate 80386+

Может принимать 16 значений, но в IDT допустимо 5: 5(задачи), 6(прерываний 286), 7(ловушки 286), Eh(прерываний 3/486), Fh(ловушки 3/486) (это по РФ)

  - 4-0 (а вообще это S - system (0 - системный объект, 1 - обычный)
  - 5-6-DPL - уровень привилегий (0 - уровень привилегий ядра, 3 - пользовательский/приложений, *1-2 - не используется в системах общего назначения*)
  - 7-1 (а вообще это P - бит присутствия (1 - если сегмент в оперативной памяти, 0 - иначе)

Пример заполнения IDT из лабораторной работы.

```
;Структура idescr для описания дескрипторов (шлюзов) прерываний
idescr struc
    offs_l  dw 0
    sel     dw 0
    cntr    db 0
    attr    db 0
    offs_h  dw 0
idescr ends
```

```
;Таблица дескрипторов прерываний IDT
;Дескриптор: <offs_l, sel, rsrv, attr, offs_h>
;смещение позже?, селектор 32-разрядного сегмента кода
idt label byte

; Первые 32 элемента таблицы - под исключения-внутренние прерывния процессора
;(реально-18, остальные-зарезервированы)
;attr=8Fh: тип=ловушка 386/486(обр. программные пр. и искл., IF не меняется),
;системный объект, УП ядра, P=1
idescr_0_12 idescr 13 dup (<0,code32s,0,8Fh,0>)
; исключение 13 - нарушение общей защиты (нарушение, код ошибки-та команда)
; происходит: за пределами сегмента, запрет чтения, за гр. таблицы дескр., int с отс. номером
idescr_13 idescr <0,code32s,0,8Fh,0>
idescr_14_31 idescr 18 dup (<0,code32s,0,8Fh,0>)


; Затем 16 векторов аппаратных прерываний,
;attr=8Eh: тип=прерывание 386/486(обр. аппаратные пр., IF сбрасывается а iret восстанавливает),
;системный объект, УП ядра, P=1
;Дескриптор прерывания от таймера
int08 idescr <0,code32s,0,8Eh,0>
int09 idescr <0,code32s,0,8Eh,0>

idt_size = $-idt        ;размер
ipdescr df 0            ;псевдодескриптор
ipdescr16 dw 3FFh, 0, 0 ;содержимое регистра IDTR в РР: с адреса 0, 256*4=1кб=2^10
```

```
; Заносим в дескрипторы прерываний (шлюзы) смещение обработчиков прерываний.
lea eax, es:except_13
mov idescr_13.offs_l, ax
shr eax, 16
mov idescr 13.offs h, ax
```

2 билет

2.1 Классификация операционных систем. Особенности ОС определенных типов. Виртуальная машина и иерархическая машина – декомпозиция системы на уровни иерархии, иерархическая структура Unix BSD, архитектуры ядер ОС – определение, примеры.

Классификация операционных систем. Особенности ОС определенных типов

1. Однопрограммная пакетной обработки

В оперативной памяти может быть только 1 прикладная программа. Режим работы ЭВМ: однопрограммный/однозадачный.

2. Мультипрограммная пакетной обработки

В оперативной памяти одновременно много программ. Загрузка – перфокартами. Программа располагается в памяти целиком; процессорное время выделяется по принципу приоритетов. Планирование на принципе распараллеливания функций. Программа выполняется до тех пор, пока не запросит доп. ресурс системы или пока не придет более высокоприоритетный процесс, в случае поддержки вытеснения, может вытеснить. Режим работы ЭВМ: мультипрограммный/мультизадачный/многозадачный

3. Системы разделения времени

В оперативной памяти одновременно находится большое число программ, процессорное время квантуется. Процесс выполняется до тех пор, пока не истек квант, не начался

процесс ввода/вывода или не вытеснен другим высокоприоритетным процессом. Режим работы ЭВМ: однопрограммный или мультипрограммный

4. Системы реального времени

POSIX определение реального времени в ОС (стандарт 1003.1

Реальное время в операционных системах – это способность операционной системы обеспечить требуемый уровень сервиса в определенный промежуток времен, величина которого определяется особенностями работы внешнего по отношению к вычислительной системе устройства или процесса. Время отклика системы <= время поступления запроса на ответ.

В отличие от систем общего назначения ОСРВ обеспечивает ответ системы (или сервис системы) за определенный промежуток времени, то есть обслуживание запроса. Это всегда запрос или внешний по отношению к системе или внешнего процесса. 2 процесса реального времени в наших компьютерах – видео и аудио.

Жесткое реальное время – это когда интервал установлен жестко и не может быть превышен. Мягкое реальное время – возможность небольших отклонений от величины интервала.

Пакет – набор программ, которые одновременно загружены в память.

И еще вроде

5. *серверные ОС – предоставляющие доступ к аппаратным (принтеры) и программным ресурсам (файлы доступа к Интернет) из сети.*
6. *многопроцессорные*
7. *встроенные ОС (телевизоры, микровольновые печи).*
8. *опер системы для смарт-карт (самые маленьеие операционные системы).*
9. *Персональные операционные системы*

В многозадачных ОС (операционных многозадачных системах пакетной обработки и системах разделения времени) в состоянии готовность одновременно может быть большое количество процессов, они организуют очередь, а в однозадачных (DOS, однозадачной пакетной обработки) – только один.

Дисциплины планирования связаны с соответствующими типами ОС (к однозадачным не относится).

Два вида: мультизадачные системы пакетной обработки и системы разделения времени (мультизадачные во втором случае нельзя говорить)

В системах пакетной обработки пользователь отделен от процесса выполнения программы. Рисунок (человек {пакет} квадрат). В системах разделения времени – наоборот: пользователь непосредственно связан – запускает, вводит данные, получает результат и реагирует – интерактив.

Иерархическая структура unix BSD 4.4. На самом нижнем – диспетчеризация – выделение процессам процессорного времени (в срв или пакетной обработки)

Виртуальная машина и иерархическая машина – декомпозиция системы на уровни иерархии

Виртуальная машина – кажущаяся возможной. Виртуальная машина – набор команд и функций, необходимых пользователю для получения сервиса операционной системы

Иерархическая машина (Медника-Донована.

ОС разбивается на функции и определяется место этих функций по удаленности от аппаратной части.

Между уровнями определяется интерфейс взаимодействия. Интерфейс – функции, которые *нижние уровни предоставляют верхним уровням.*

- прозрачные - позволяет обращаться через уровни
- полупрозрачные - какие-то обращения через уровни возможны (какие-то только к низлежащим уровням)
- непрозрачные - обращения возможны только к низлежащим уровням

- Управление процессами (нижний уровень) – выделение процессу процессорного времени.
- уровень 2 - (P, V) - семафорные операции, контр. доступа к разделяемым ресурсам, планировщик процессов (операция более высокого уровня. постановка процессов в очередь к процессору или другим ресурсам), управление памятью; Процессор, оперативка, внешние устройства - аппаратные устройства
- Управление процессами (верхний уровень) – создание, уничтожение, взаимодействие при помощи сообщений. Система принимает решение об этом, более высокоуровневая операция
- уровень 4 - управление устройствами (подсистема ввода/вывода);
- уровень 5 - управление информацией (файловая система). Система предоставляет такие средства как наименование файлов, построения дерева каталогов, удобных для пользователя.

Виртуальная машина - совокупность команд машины и команд ОС, которые могут использовать программы для получения сервиса ОС.

Иерархическая структура unix BSD 4.4

Таблица 1: Структура ядра ОС UNIX 4.4 BSD

<table>
<tr><td colspan="3">Системные вызовы</td><td colspan="3">////////////</td><td>Сис. выз.</td><td colspan="2">////////////////</td></tr>
<tr><td colspan="3">Управление терминалом</td><td>1</td><td>2</td><td>3</td><td>4</td><td>Сокеты</td><td rowspan="2">Обработка сигналов</td><td rowspan="2">Создание и завершение процесса</td></tr>
<tr><td rowspan="2">Необра-ботан-ный телетайп</td><td>Обработанный телетайп</td><td colspan="2">Файловая система</td><td colspan="2">Виртуальная память</td><td>Сетевые протоколы</td></tr>
<tr><td>Дисциплины линий связи</td><td colspan="2">Буферный КЭШ</td><td colspan="2">Страничный КЭШ</td><td>Маршру-тизация</td><td colspan="2">Планирование процессов</td></tr>
<tr><td colspan="3">Драйверы символьных устройств</td><td colspan="4">Драйверы блочных устройств</td><td>Драйверы сетевых устройств</td><td colspan="2">Диспетчеризация процессов</td></tr>
</table>

1. Символьный уровень

2. Именование

3. Отображение адреса

4. Страничное прерывание

/ - аппаратные и эмулируемые прерывания

Юникс хорошо структурирована, видим в части процессов все то же, что у медника-денована (там только не было сигналов). Но в Юникс сигналы информируют процессы о событиях, которые происходят в системе – очень важно.

На самом нижнем – диспетчеризация – выделение процессам проц времени (в срв или пакетной обработки)

Более высокий уровень – планирование – определение, в какой последовательности будут выполняться процессы. Если реализуется приоритетное планирование (при этом учет времени простоя в очереди готовых процессов) – приоритеты будут динамическими и изменяться в зависимости от времени простоя (и бывает от количества полученного процессорного времени). То же делают и винды, но не по формуле, а просто сканируя.

Еще более высоко – создание и завершение процессов. Порождение – идентификация, выделение и инициализация дескриптора. Завершение – действие, в результате которого все занимаемые процессом ресурсы возвращаются системе.

Также на нижнем уровне находятся драйверы – ПО, которое напрямую взаимодействует с аппаратной, предназначено для управления внешними устройствами.

Два типа устройств – символьные (сетевые в тд) и блочные (магнитные диски, одна из функций – поддержка виртуальной памяти – совокупность всех АП процессов, которые в данный момент выполняются на компе).

Страничный и буферный кеш (относится к файловой системе, так как все буферизуется).

архитектуры ядер ОС – определение, примеры.

Существует два класса ядер:

1. Монолитные ядра (более ранняя)

2. Микроядра

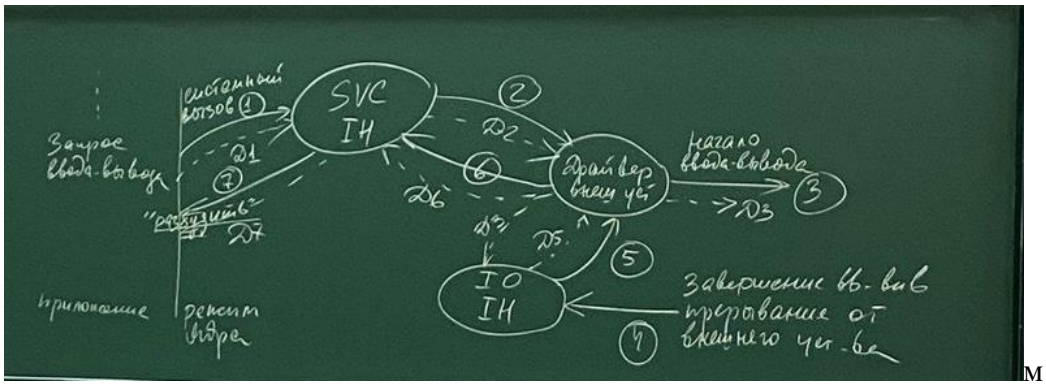
Монолитное ядро

Это программа, имеющая модульную структуру, то есть состоящая из подпрограмм. Системы Windows, Unix, Linux имеют монолитные ядра. Unix имеет минимизированное ядро (вынесен графический интерфейс).

В архитектуре с монолитным ядром все услуги для прикладного приложения выполняют отдельные части кода ядра (в адресном пространстве ядра)

Все построено на прерываниях – системные вызовы, исключения и аппаратные прерывания.

Последовательность действий в системе при запросе приложения на ввод/вывод



Происходит вызов read/write и система переходит в режим ядра (ни одна система не дает прямое обращение к внешним устройствам)

1. Системный вызов из функции, который переводит в режим ядра, туда же соответствующие данные. Super visor call . Супервизор - … в стадии выполнения. IH- interrupt handler.
2. В результате обработки системного вызова будет вызван драйвер внешнего устройства (программа ввода/вывода). Драйверу будут переданы данные в его формате.
3. Драйвер инициализирует работу внешнего устройства. На этом управление процессором работы заканчивается, он отключается, потому что работой внешнего устройства управляют контроллеры.
4. По завершении операции ввода/вывода будет сформировано контроллером устройства прерывание, которое в прост схеме поступит на контр прерывания и в результате будет определен адрес точки входа обработчика прерывания,
5. И обработчик начнет выполняться (IO IH). Процессор перейдет на выполнение обработчика, тк аппаратные прерывания имеют наивысший приоритет. ОП входят в состав драйвера и является одной из точек входа драйвера внешнего устройства. Драйвер всегда содержит 1 обработчик прерывания.
6. Поскольку обработчик – точка входа драйвера, у драйвера есть call back функция – задача вернуть запрашиваемые данные приложению. В результате драйвер через подсистему ввода/вывода должен передать данные приложению.
7. Для этого работа приложения дб возобновлена. Процесс, который запросил вв блокируется. (7) – разбудить.

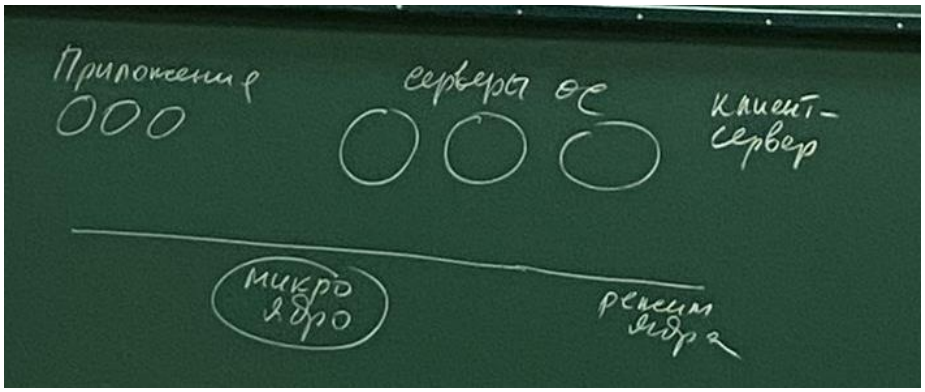
С монитором мы работаем как с памятью – mov. А input/output – использование команд ввода вывода, и это приводит к блокировкам.

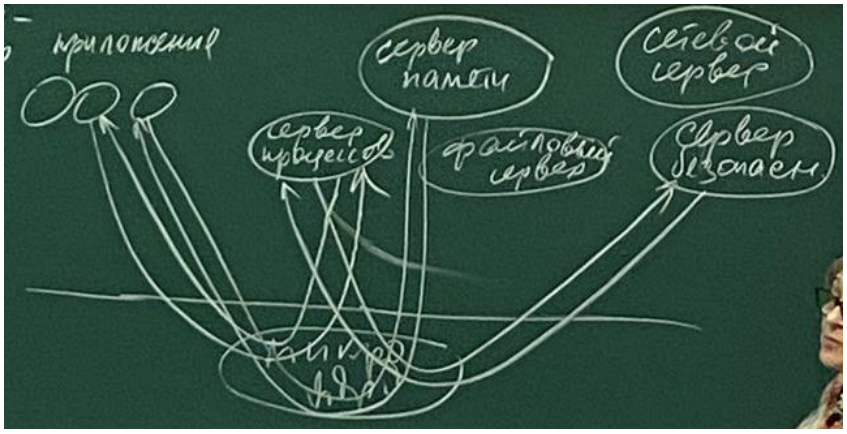
Возникновение прерывания происходит асинхронно. Чтобы получить значение, процесс разблокируется. поэтому эта схема называется блокирующий синхронный ввод-вывод

Микроядерная архитектура

Mach – первая ОС с микроядром. Классическим примером микроядерной ОС является Symbian OS.

Идея мя – оставить в ядре только самые низкоуровневые функции, остальные функции выполняются в режиме пользователя. Поскольку остальная часть ОС реализуется в виде отдельных процессов в собственных АП, то взаимодействие между компонентами ОС выполняются с помощью посылки и приема сообщений, причём этот механизм обеспечиваются специальным модулем ядра, который называется микроядро.





Современные ОС предоставляют возможность внесения в монолитное ядро собственных … без перекомпиляции. В Unix – загружаемые модули ядра. С помощью них можно сделать не все. Если надо изменить структуру ядра – надо перекомпилировать. Поэтому идея микроядра – вносить изменения в ОС, имея широкие возможности и не перекомпилировать.

Программы ОС принято называть серверами ОС. Суть – та же. Монолитное ядро предоставляет приложениям сервис (с помощью системных вызовов).

Взаимодействие микроядреной архитектуры по модели клиент-сервер.

Модель предполагает, что программы обращаются к серверу с к-либо запросами, эти запросы сервер обрабатывает, в результате формируется ответ, ожидаемый клиентом. Любая такая архитектура предполагает взаимодействие по протоколу – соглашение.

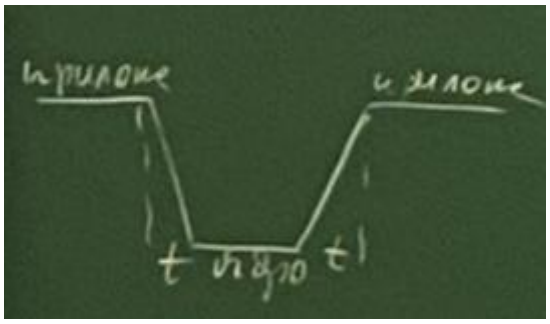
Такое взаимодействие дб надежным – посылка сообщения должна подтверждаться сообщением о его приеме. В результате мы рассматривали диаграмму – 3 состояния блокировки при передаче сообщений. Все 3 состояния будут присутствовать – блокирован при посылке, блокирован при ответе, блокирован при приеме.

Это очень важная диаграмма состояний, напрямую связана с эффективностью микроядерной архитектуры.

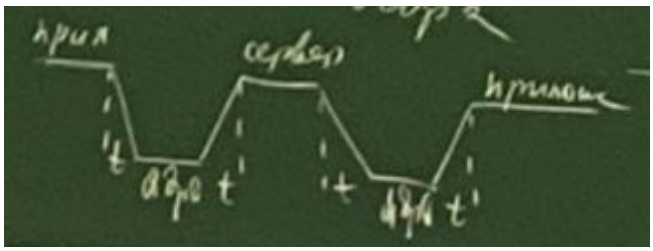
Mach – первая такая ОС

Эффективность

В монолитном ядре при обработке системного вызова будет выполнено 2 переключения – 1 из режима задачи в режим ядра, 2 – обратно.



В мя – минимум 4



Но есть вероятностные временные затраты – блокировки при приеме и ответе – которые оценить невозможно.

Несмотря на то, что эффективность мя архитектуры намного ниже, интерес не утрачивается. В чем же привлекательность.

В том, что большая часть кода ядра вынесена в режим пользователя и мб изменена без перекомпиляции ядра.

Но как пишут соломон с русиновичем, для коммерческих реализаций мя mach перестает быть уже таким микро. В частности, файловая подсистема, поддержка сетей и управление памятью в коммерческих системах mach выполняется в режиме ядра как в системах с монолитным ядром.

Причина проста: системы, построенные строго по принципу мя плохи с коммерческой тз из-за низкой эффективности. Но мя используется в системах реального времени. Например, широко известная срв QNX построена на основе мя архитектуры.

2.2 Три режима работы вычислительной системы с архитектурой X86: особенности. Реальный режим: линия A20 – адресное заворачивание. Перевод компьютера в защищенный режим. Линия A20 в защищенном режиме: включение и выключение линии A20 (код из лабораторной работы).

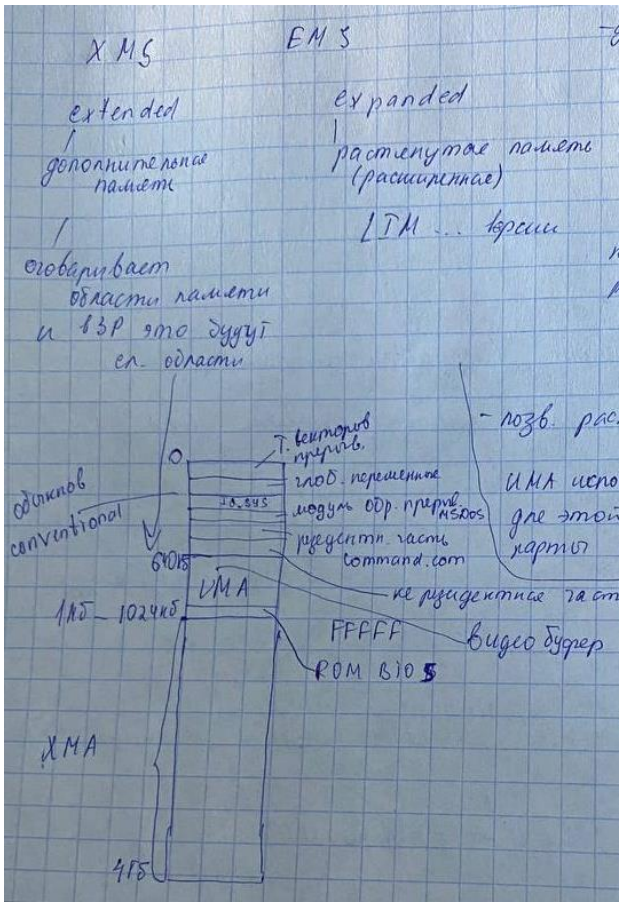
Три режима

Линия a20

Рассмотрим 2 спецификации (в России – госты).

XMS	EMS
extended memory specification	expanded memory specification
Дополнительная память	Растянутая (расширенная) память
	У нее много версий, начинаются словами LIM…
Оговаривает область памяти и в защищенном режиме это будут следующие области	Что-то там … страниц Это позволяет растянуть память. Это называется виртуализация и мы получаем дополнительный что-то там…

С появлением 8086 стала доступна память 1МБ – upper memory area. Использовалась в 8086 под управлением dos, для нее были опубликованы карты первого мбайта. Память сверх одного мбайта –XMA. В защищенном режиме 32 р шина адреса – смещения могут достигать 4 гб – это 8 ф FFFFFFFF



адресное заворачивание

В РР было 20 адресных линий и 2 типа адресного заворачивания:

- Если у нас 20 единиц – 5 ф и мы прибавили еще 1, то все выльется, «со стола не слижешь», становятся 5 нулей и адрес начинает указывать на младшие адреса.
- Второй тип – в памяти сегменты. Максимально смещение 64Кбайт: Ffff+1=0000 – указывает на младшие адреса сегмента.

Нас интересует первое.

Для обеспечения обратной совместимости (аппаратно поддерживает обратную совместимость) создан порт линии а20. У нас 32 адресные линии, поэтому это уже не 20 линия.

В реальном режиме линия а20 закрыта. Она на 0 потенциале, заземлена. Когда переводим в зр, ее надо открыть. Что будет, если забудем - то получаем битую память. Адреса с 1 в этом бите нам не будут доступны. При переходе обратно, надо закрыть. Если не закроем – она нам будет доступна.

НЮАНС

Если мы в реальном режиме на компы 32 или 64 разрядные – можем поставить dos и комп будет работать под его управлением.

Если мы в реальном режиме откроем линию а20, нам станет доступно еще 64 кбайт памяти- high memory area. Но это в реальном режиме. В защищенном режиме нам доступно 4 ГБ.

«На хабрах ваших вонючих» пишут про теневые регистры еще. РФ пишет ffff в теневые регистры, но старшие 4 разряда не обнуляют. «Идиоты». Если остались действительно 4 гб сегменты и оставить, то будет доступно 4 гб – «жуть и чушь». Как мы будем их адресовать??

То есть если не запишем в limit ffff – криминала никакого.

включение и выключение линии А20

Существует 2 кода, которые используют при работе с а20:

Открыть

```
Mov al, 0D1h ; команда управления линией а20

Out 64h, al

Mov al, oDFh ; код открытия линии а20

Out 60h, al
```

или

```
In al, 0x92

Or al, 2

Out 0x92, al
```

Начиная с PC/2 имеется быстрый (2) вариант открытия линии a20. Он исключает опрос. Быстрый вариант считается не вполне надежным и поддерживается не всеми платформами. Невозможно убедиться в этом заранее. Поэтому пользоваться надо 1 вариантом.

Закрыть линию a20. Та же команда управления

Mov al, 0D1h ; команда управления линией a20

Out 64h, al

Mov al, oDDh; код закрытия линии a20

Out 60h, al

Перевод компьютера в защищенный режим

Чтобы перейти в защищенный режим, достаточно установить бит PE - нулевой бит в управляющем регистре CRO, и процессор немедленно окажется в защищенном режиме. Единственное дополнительное требование, которое предъявляет Intel, - в этот момент все прерывания, включая немаскируемое, должны быть отключены (Зубков, стр. 488).

1. Подготовить в оперативной памяти глобальную таблицу дескрипторов GDT. В этой таблице должны быть созданы дескрипторы для всех сегментов, которые будут нужны программе сразу после того, как она переключится в защищённый режим. Впоследствии, находясь в защищённом режиме, программа может модифицировать GDT (если, она в нулевом кольце защиты).
2. Подготовить в оперативной памяти таблицу дескрипторов прерываний IDT.
3. Записать линейные физическе адреса в дескрипторы сегментов.
4. Загрузить адрес и размер GDT в GDTR.
5. Загрузить смещения обработчиков прерываний в шлюзы.
6. Сохранить маски прерываний.
7. Перепрограммировать ведущий контроллер прерываний на новый базовый вектор (32).
8. Запретить все маскируемые и немаскируемые прерывания.
9. Открыть адресную линию A20.
10. Загрузить адрес и размер IDT в IDTR.
11. Перейти в защищенный режим (установить бит PE — нулевой бит в управляющем регистре CR0 в 1).
12. Загрузить новый селектор в регистр CS.
13. Загрузить сегментные регистры селекторами на соответствующие дескрипторы.
14. Разрешить прерывания.

3 билет

3.1 Прерывания: классификация. Последовательность действий при выполнении запроса ввода-вывода. Обработчики аппаратных прерываний: виды и особенности. Функции обработчика прерываний от системного таймера в ОС семейства Windows и семейства Unix.

3.2 Защищенный режим: назначение системных таблиц – глобальной таблицы дескрипторов (GDT), таблицы дескрипторов прерываний (IDT), теневых регистров (структуры, описывающие дескрипторы GDT и IDT и заполнение дескрипторов в лабораторной работе по защищенному режиму). Адресация памяти в ЗР

4 билет

4.1 Тупики: Обнаружение тупиков для повторно используемых ресурсов методом редукции графа, способы представления графа, алгоритмы обнаружения тупиков. Пример анализа состояния системы метод редукции графа. Методы восстановления работоспособности системы.

4.2 Задача «Обедающие философы» – модели распределение ресурсов вычислительной системы. Множественные семафоры UNIX: системные вызовы, поддержка в системе, пример использования из лабораторной работы «производство-потребление».

5 билет

5.1 Виртуальная память: распределение памяти страницами по запросам, схема с гиперстраницами, обоснование использования данной схемы. Управление памятью страницами по запросам в архитектурах x86 – расширенное преобразование (PAE) – схема преобразований. Анализ страничного поведения процессов: свойство локальности, рабочее множество.

5.2 Задача «Производство-потребление»: алгоритм Эд. Дейкстры, реализация на семафорах UNIX (код из лабораторной работы)

6 билет

6.1 Понятие процесса. Процесс как единица декомпозиции системы. Диаграмма состояний процесса с демонстрацией действий, выполняемых в режиме ядра. Планирование и диспетчеризация. Классификация алгоритмов планирования. Примеры алгоритмов планирования, соотнесенные с типами ОС. Процессы и потоки. Типы потоков.

6.2 Обеспечение монопольного доступа к разделяемым данным в задаче «читатели-писатели» : реализация на базе Win32 API (пример кодов лабораторной работы «читатели-писатели» для ОС Windows).

7 билет

7.1 Управление виртуальной памятью: распределение памяти сегментами по запросам: схема преобразования виртуального адреса, способы организации таблиц сегментов, стратегии выбора разделов памяти для загрузки сегментов, алгоритмы и особенности замещения сегментов.

7.2 Управление памятью сегментами по запросам в архитектуре X86. Тип организации таблиц сегментов. Формат дескриптора сегмента в таблицах дескрипторов сегментов (GDT и LDT) (код и заполнение дескрипторов GDT из лабораторной работы по защищенному режиму).

8 билет

8.1 Взаимоисключение и синхронизация процессов и потоков. Семафоры: определение, виды. Семафор, как средство синхронизации и передачи сообщений. Семафоры UNIX: примеры решения задач с помощью семафоров: «Производство-потребление» и «Читатели-писатели» в UNIX (пример реализации в лабораторной работе).

8.2 Аппаратные прерывания: задачи обработчика прерываний от системного таймера в защищенном режиме.

9 билет

9.1 Виртуальная память: управление памятью страницами по запросу – три схемы преобразования; реализация страничного преобразования в компьютерах на базе процессоров Intel (x86): стандартное преобразование и PAE в защищенном режиме – схемы, размеры таблиц и их количество на каждом этапе преобразования.

9.2 Unix: концепция процессов; иерархия процессов, процессы «сироты», процессы «зомби», демоны; примеры из лабораторной работы (5 программ).

10 билет

11 билет

11.1 Параллельные процессы: взаимодействие, обоснование необходимости монопольного доступа к разделяемым переменным, способы взаимоисключения. Мониторы: определение; примеры – простой монитор и монитор кольцевой буфер.

11.2 Средства межпроцессорного взаимодействия (IPC) операционной системы UNIX System V: очереди сообщений и программные каналы – сравнение, примеры (для программных каналов пример из лабораторной работы с сигналами).

12 билет

12.1 ОС с монолитным ядром. Переключение в режим ядра. Диаграмма состояний процесса и переход из одного состояния в другое – причины каждого перехода. Диаграмма состояний процесса в UNIX. Переключение контекста. Система прерываний.

12.2 Задача: читатели-писатели – монитор Хоара, решение с использованием семафоров Unix и разделяемой памяти, пример реализации из лабораторной работы.

13 билет

13.1 Виртуальная память: управление памятью страницами по запросу – три схемы. Алгоритмы вытеснения страниц: демонстрация особенностей на модели траектории страниц. Рабочее множество – определение, глобальное и локальное замещение. Флаги в дескрипторах страниц, предназначенные для реализации замещения страниц.

13.2 Синхронизация и взаимоисключение параллельных процессов в распределенных системах: централизованный и распределенный алгоритмы, алгоритмы Token-ring; сравнение алгоритмов. Транзакции: определение, особенности, двухфазный протокол фиксации.

14 билет

14.1 Процессы: взаимодействие процессов в распределенных системах; централизованный и распределенный алгоритмы, синхронизация часов (алгоритм Лампорта); RPC – механизм.

14.2 Аппаратные прерывания: типы аппаратных прерываний; особенности. Прерывания от устройств ввода-вывода: назначение и аппаратная реализация. Прерывание от системного таймера в защищенном режиме. Пример кода обработчика прерывания от системного таймера из лабораторной работы по защищенному режиму.

15 билет

15.1 Межпроцессорное взаимодействие в Unix System V (IPC): сигналы, программные каналы, семафоры и разделяемая память; примеры использования из лабораторных работ. 15.2 Синхронизация и взаимоисключение параллельных процессов в распределенных системах: централизованный и распределенные алгоритмы – сравнение.

16 билет

16.1 Взаимодействие параллельных процессов: проблемы; монопольный доступ и взаимоисключение; взаимодействие параллельных процессов в распределенных системах – особенности; централизованный алгоритм, распределенный алгоритм; синхронизация логических часов (алгоритм Лампорта).

16.2 Процессы в UNIX: системные вызовы fork(), exec(), wait(), signal() – примеры из лабораторных работ.

17 билет

17.1 Виртуальная память: распределение памяти страницами по запросам, свойство локальности, рабочее множество, анализ страничного поведения процессов. Схема страничного преобразования в процессорах Intel (X86) PAE – размеры таблиц дескрипторов.

17.2 Прерывания от системного таймера в защищенном режиме: функции (по материалам лабораторной работы).

18 билет

18.1 Процессы: взаимодействие параллельных процессов – монопольный доступ и взаимоисключение; программная реализация взаимоисключения – флаги, алгоритм Деккера, алгоритм Лампорта.

18.2 Защищенный режим: перевод компьютера в защищенный режим – реализация – пример кода из лабораторной работы.

19 билет

19.1 Процессы Unix: создание процесса в ОС Unix и запуск новой программы. Примеры программ из лабораторных работ, демонстрирующих эти действия. Системные вызовы wait() и pipe(): назначение, примеры из лабораторных работ. Процессы «сироты», «зомби» и «демоны».

19.2 Взаимодействие параллельных процессов: мониторы – определение; монитор Хоара «читателиписатели», реализация в для ОС Windows – пример из лабораторной работы. .

20 билет

20.1 Процессы: взаимодействие параллельных процессов – монопольный доступ и взаимоисключение; программная реализация взаимоисключения – примеры, семафоры – определение, виды семафоров, примеры использования множественных семафоров из лабораторных работ «производство-потребление» и «читатели-писатели».

20.2 Приоритетное планирование в ОС Windows (лабораторная работа).

21 билет

21.1 Взаимодействие параллельных процессов: монопольное использование – реализация; типы реализации взаимоисключения. Мониторы – определение, примеры: простой монитор, монитор «кольцевой буфер» и монитор «читатели-писатели». Пример реализации монитора Хоара «читатели-писатели» для ОС Windows.

21.2 Процессы Unix: создание процесса в ОС Unix и запуск новой программы. Примеры из лабораторной работы (код).

22 билет

22.1 Процессы: взаимодействие параллельных процессов – монопольный доступ и взаимоисключение; аппаратная реализация взаимоисключения, спин-блокировка – реализация.

22.2 Процессы: бесконечное откладывание, зависание, тупиковая ситуация – анализ на примере задачи об обедающих философах и примеры аналогичных ситуаций в ОС. Множественные семафоры в Linux: системные вызовы и поддержка в ОС Linux; примеры из лабораторных работ.

23 билет

24 билет

24.1 Процессы: взаимодействие параллельных процессов – монопольный доступ и взаимоисключение; алгоритм Лампорта «Булочная» и «Логические часы» Лампорта.

24.2 Процессы: бесконечное откладывание, зависание, тупиковая ситуация – анализ на примере задачи об обедающих философах и примеры аналогичных ситуаций в ОС. Множественные семафоры в Linux: системные вызовы и поддержка в системе; пример из лабораторной работы «производство-потребление». .

25 билет

26 Дополнительные вопросы

26.1 XMS

26.2 Методы организации ввода-вывода: программируемый, с прерываниями, прямой доступ к памяти.

26.3 Кэши TLB и данных

26.4 ОС с монолит. ядром. Переключение в режим ядра. Система прерываний. Точные и неточные прерывания.

26.5 Спецификация XM (XMS): Conventional, HMA, UMA, EMA.

26.6 Управление памятью: выделение памяти разделами фиксированного размера, выделение памяти разделами переменного размера, стратегии выделения памяти, фрагментация памяти.

26.7 Тупики: Обнаружение тупиков для повторно используемых ресурсов методом редукции графа, способы представления графа и методы восстановление работоспособности системы

26.8 Последовательность операций при выполнении аппаратного прерывания. Прерывания точные и неточные

26.9 EMS .

26.10Классификация структур ядер ОС. Особенности ОС с микроядром. Модель клиент-сервер. Три состояния процесса при передаче сообщений. Достоинства и недостатки микро-ядерной архитектуры.

26.11Тупики: определение тупиковой ситуации для повторно используемых ресурсов, четыре условия возникновения тупика, обход тупиков - алгоритм банкира.
26.12Прерывание int 8h (реальный режим) - функции.