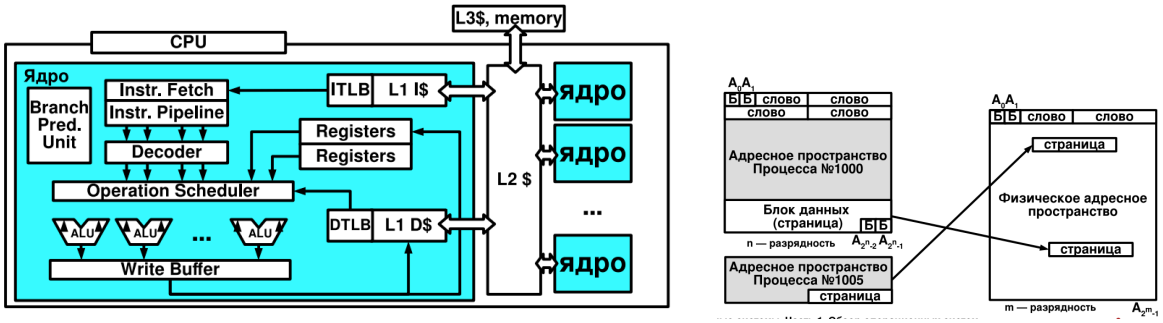
1. Архитектура компьютерных систем. Архитектура Фон-Неймана и Гарвардская

Архитектура Фон-Неймана: память используется и для инструкций и для данных, единая шина передачи данных и инструкций. (Более просто, универсально, bottleneck шина, замедляет работу)
Гарвардская: две раздельные памяти для инструкций и для данных, две раздельных шины. (Более сложно, но быстрее)
NUMA/UMA: архитектуры организации доступа процессоров к памяти
Uniform Memory Access: у всех процессоров одинаковый доступ к памяти, легко но плохо масштабируется
Non Uniform Memory Access: память разделена на блоки по процессорам, быстрее но сложнее

2. Общая организация процессора, памяти, организация вычислений.



Устройство процессора: для каждого ядра L1 кеш, для всех ядер L2 кеш, доступ к L3 через L2. В каждом ядре: L1 кеш, Instruction Fetch & Pipeline для обработки инструкций, декодер инструкций, планировщик операций, АЛУ, регистры, буферы записи, устройство предсказания ветвлений.
Организация виртуальной памяти: для каждого процесса - свое адресное пространство, память разбивается на страницы фиксированного размера. Страницы из виртуальной память сопоставляются с физической.
Организация вычислений: Ядро выполняет последовательно каждую команду Instruction Fetch => Decode => Execution => MEM => Write

3. Организация прерываний, типы прерываний, контроллер прерываний.

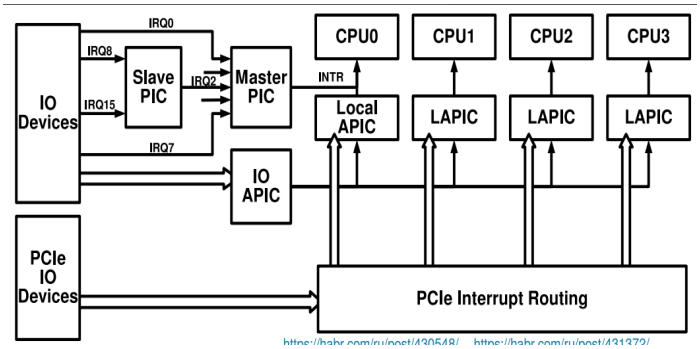
Прерывания выполняются в конце цикла команды, могут быть вложенными и иметь приоритеты.

Типы прерываний:

Синхронные: (как результат выполнения программы): Syscalls, Pagefaults, Ошибки
Асинхронные: (внешние события) IO Interrupts, Time Interrupts, Kernel events

Контроллер прерываний:

Устройство посылает сигнал прерывания
IO APIC решает как обрабатывать прерывание и посылает его на LAPIC
LAPIC передает управление ядру
Процессор выполняет обработчик



4. Типичные функции операционной системы. Интерфейсы ОС. Работа ОС как замена оператора ЭВМ.

Функции ОС: выполнение программ, доступ к io, контролируемый доступ к файлам, доступ к системе и ресурсам, обработка ошибок, учет и диспетчеризация ресурсов
Интерфейсы:
Instruction Set Architecture набор команд
Application Binary Interface бинарный интерфейс приложений
Application Programming Interface интерфейс прикладных программ
Замена оператора ЭВМ:
Раньше оператор ЭВМ должен был получить программу, вбить ее на перфокартах, загрузить ее и компилятор, запустить и передать распечатку.

5. Пакетная обработка. Системный монитор.

Пакетная обработка: что бы избежать простои машинного времени (очень дорогого раньше) придумали метод, где программы и данные собираются в пакеты и обрабатываются последовательно
Системный монитор: предшественник ОС, включает в себя обработчик прерываний, драйверы устройств, планировщик заданий, интепретатор

6. Анализ общесистемной эффективности, как предусловие многозадачности. Многозадачность, как способ повышения системной эффективности. Системы разделения времени.

Так как чтение и запись гораздо более затратны по времени, мы пришли к проблеме что, одно задание, испольщующее ввод-вывод плохо загружает CPU и дорогое машинное время расходуется просто так. Человечество пришло к запуску нескольких задач и пока одни будут заниматься вводом-выводом, другие будут заниматься вычислениями
Далее были придуманы системы разделения времени, позволяющие исключить операторов. Пользователи стали пользоваться компьютером, им выдавалась часть процессорного времени.

7. Процессы, проблемы современных процессов. Планирование выполнения процессов и управление ресурсами.

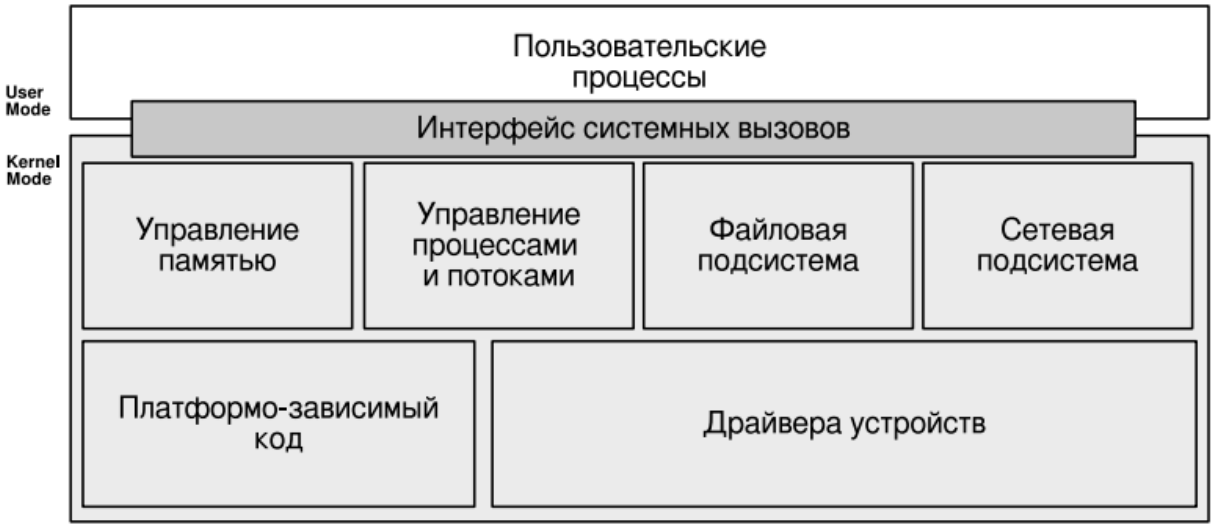
Процесс - единица активности ОС в которой последовательные действия, текущее состояние и связанные ресурсы
Структура процесса:
Исполняемая программа, набор потоков исполнения, связанные структуры ядра, адресное пространство (стек куча данные), контекст исполнения, контекст безопасности, ресурсы, динамические библиотеки
Проблемы процессов:
Защита памяти процессов (программы могут влиять на общие куски памяти внося неопределенность), взаимные блокировки deadlocks/starvation/livelock, проблемы синхронизации, взаимное исключение доступа к ресурсам
Планирования: равноправие между пользователями, общесистемная эффективность, дифференциация отклика, Time Sharing, Interactive, Real Time, System, Fair Share, Fixed

8. Управление памятью, виртуальная память. Защита информации и безопасность ОС.

Для каждого процесса создается отдельное виртуальное адресное пространство.
Так же есть
Управление памятью: heap allocator, kernel allocator (для компонентов ядра, slab для небольших объектов, buddy для больших, mapping files - отображение файлов как страниц)
Долгосрочное хранилище - хранение во вторичной памяти
Поддержка модулей, динамическая загрузка
Защита и контроль доступа к различным областям памяти

Paging/Swapping - механизм при котором страницы перемещаются из оперативной памяти во вторичное хранилище и обратно.
Файл подкачки - туда сгружаются страницы с оперативной памяти
MMU - аппаратный компонент преобразования виртуальных адресов в физические
TLB кеш внутри MMU
Защита: разделение прав доступа, аутентификация и авторизация, защита от нецелостного изменения

9. Структура ядра операционной системы. Архитектуры монолитного ядра, ядра динамически загружаемыми модулями и микроядра.



По архитектуре ядра делятся на:
Монолитное: все функции ядра реализованы в одном пространстве памяти, высокая производительность, сложность дебага, не очень надежно
С динамически загружаемыми модулями: части ядра могут загружаться и выгружаться во время работы системы (драйверы или файловые системы например)
Микроядро: ядро выполняет минимальный функционал - все остальное в модулях, высокая надежность простота дебага меньше производительность

10. Потоки исполнения, многопоточность, модели многопоточности.

Поток - единица диспетчеризации и выполнения ОС (контекст процессора и часть стека)
Процесс - один или несколько потоков (связанные ресурсы)

Многопотоность - технология где процесс разделяется на несколько одновременно выполняемых потоков. Полезно для приложений выполняющих несколько независимых заданий не требующих последовательного выполнения
POSIX Threads - библиотека порождения потоков
Модели многопоточности:
One to One: на каждый user поток есть kernel поток, большие накладные расходы
One to Many: все потоки сопоставляются с одним потоком ядра и управляются на уровне пользователя без вмешательства ос
Many to Many: несколько пользовательских потоков сопоставляются с меньшим кол-во ядра, баланс между гибкостью и производительностью

11. Симметричная и ассиметричная многопроцессорная обработка.

Symmetric Multiprocessing: равноправные CPU, общий пул памяти, высокая производительность, конкуренция за ресурсы, при большом кол-ве процессоров увеличивается задержка доступа к памяти, проще управление задачами
Asymmetric Multiprocessing: один главный Master CPU управляет Slave CPU, bottleneck на главном CPU, меньше гибкости и масштабируемости

12. Виртуализация. Типы виртуализации.

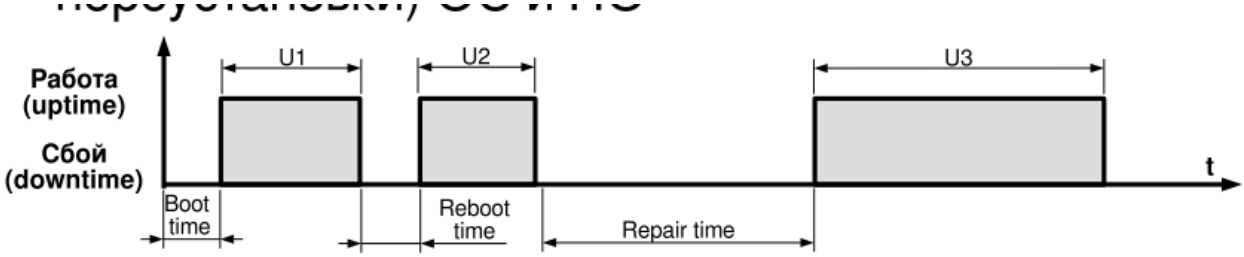
Виртуальные машины: JVM, JS, Python - машинонезависимый код
Контейнеры приложений: Docker, Linux containers (использую host kernel, меньше расходов)
Аппаратная виртуализация: KVM, Hyper-V, VMWare абстраирует физическое оборудование, максимальная изоляция, много расходов
Облачные технологии - системы построенные на базе аппаратной виртуализации, с дополнительными настройками прав доступа и общего мониторинга, выделение ресурсов

13. Сбои и отказоустойчивость ОС. Причины появления отказов в ОС и способы борьбы с ними.

Отказоустойчивость – способность системы продолжать работу при аппаратных/программных ошибках
Отказы в ОС: Ошибки аппаратуры, ошибки оператора, физические помехи окружающей среды, ошибки программирования
Постоянные/Временные/Однократные/Периодческие
Способы борьбы с отказами: избыточность оборудования, hotswap, программная поддержка подключения/отключения компонентов, RAID

14. Надежность. Среднее время восстановления. Коэффициент доступности и время простоя.

Надежность - свойство системы сохранять во времени способность выполнять требуемые функции в заданных режимах и условиях применения.
MTTF (mean time to failure) - среднее время наработки на отказ
MTTR - среднее время восстановления
Доступность - вероятность, что система находится в рабочем состоянии
Время простоя (downtime) — интервал с момента неработоспособности сервиса до момента возобновления его работы.



Availability 1 - непрерывная работа
Availability 0.999999 высокоотказоустойчивый
Availability 0.99999 отказоустойчивый
Availability 0.9999 восстанавливаемый
Availability 0.999 высокодоступный

15. Резервирование и отказоустойчивость.

Резервирование - создание дублирующих компонентов систем в случае отказа
Отказоустойчивость - способность системы продолжать функционировать при отказе
Методы резервирования: физическая избыточность (несколько компонентов, возможность hotswap), временная избыточность (повтор вычислений), информационная избыточность (ECC, RAID)
Методы повышения отказустойчивости ОС: изоляция процессов, управление параллелизмом (разрешение блокировок), виртуализация, точки восстановления, резервные копии и тд
Как правило повышение отказоустойчивости имеет трейдофф в виде производительности и денег

16. История и развитие ОС GNU/Linux. Single UNIX Specification и POSIX

Проект GNU - 1983 год Ричард Столлман написал многие ключеые компоненты такие как компилятор GCC, утилиты командной строки, текстовые редакторы

Цель - опенспорсная альтернатива UNIX
В 1991 году Линус Торвальдс начал разработку ядра **Linux**
После написания ядра **Linux** + GNU образовали полноценную ОС
В 1993 году появились первые дистрибутивы - Debian и Slackware
POSIX - набор стандартов для обеспечения совместимости между различными UNIX системами. Системные вызовы, инструменты командной строки, потоки, сигналы межпроцессорного взаимодействия.
SUS - стандарт от The Open Group для стандартизации UNIX систем, включает в себя POSIX, но дополняет его более строгими требованиями к ядру операционной системы, файловой системы и утилитам, так же сетевые функции международная поддержка и расширенные api

17. Понятие дистрибутива, дистрибутивы Linux

Дистрибутив включает в себя следующее:
Ядро, Окружение, менеджер пакетов и обновлений, графическая подсистема, прикладные программы, поддержка.
Дистрибутивы делятся на коммерческие и community
Community не имею поддержки и развиваются сообществом, например: debian/ubuntu, fedora/centosm arch gentoo
Коммерческие имеют поддержку и как правило платные: red hat, suse, oracle, astra

18. Архитектура и основные подсистемы Linux. Linux Kernel Map.

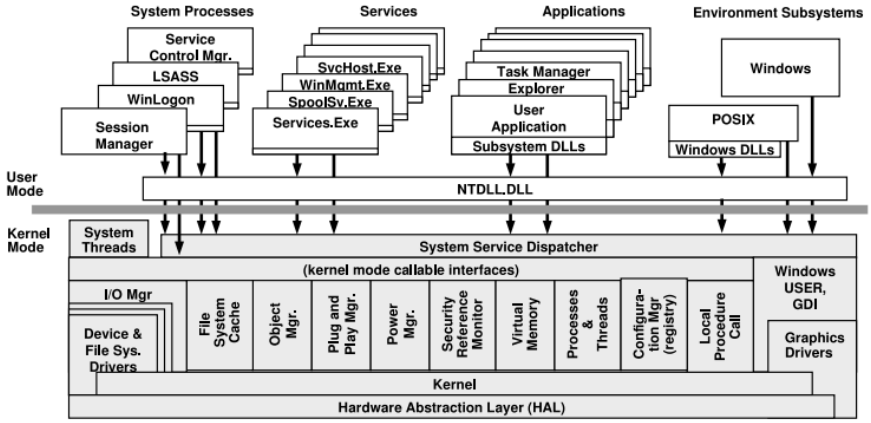
Процессы и планировщик процессов
Виртуальная память
Физическая память (управление пулом кадров страниц)
Файловая система
Драйверы символьных устройств (терминалы принтеры модемы)
Драйверы блочных устройств (различные виды вторичной памяти)
Сетевые протоколы
Драйверы сетевых устройств
Прерывания
Ловушки и отказы (обработка генерируемых процессором прерываний при сбоях)
Сигналы и IPC (управление межпроцессорным взаимодействием)

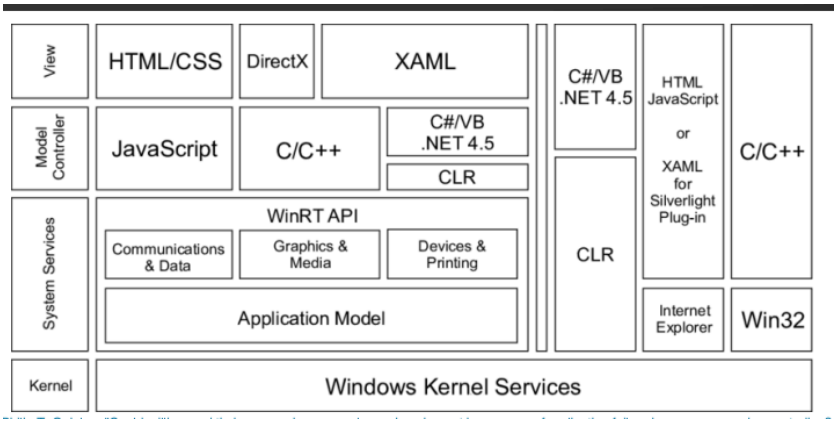
19. История и развитие Windows

Windows/MS-DOS появился как расширение примитивной операционной системы MS-DOS которая успешно пользовалась на ранних пк.
1, 2, 3, 95, 98, 2000, xp, vista, 7, 8, 10, 11

Версии 1, 2 были надстройкой над DOS
Так же есть линейки серверных версий и раньше были версии для Windows Phone, версии для IoT, для планшетов на базе arm, xbox one

20. Общая архитектура Windows. Windows API





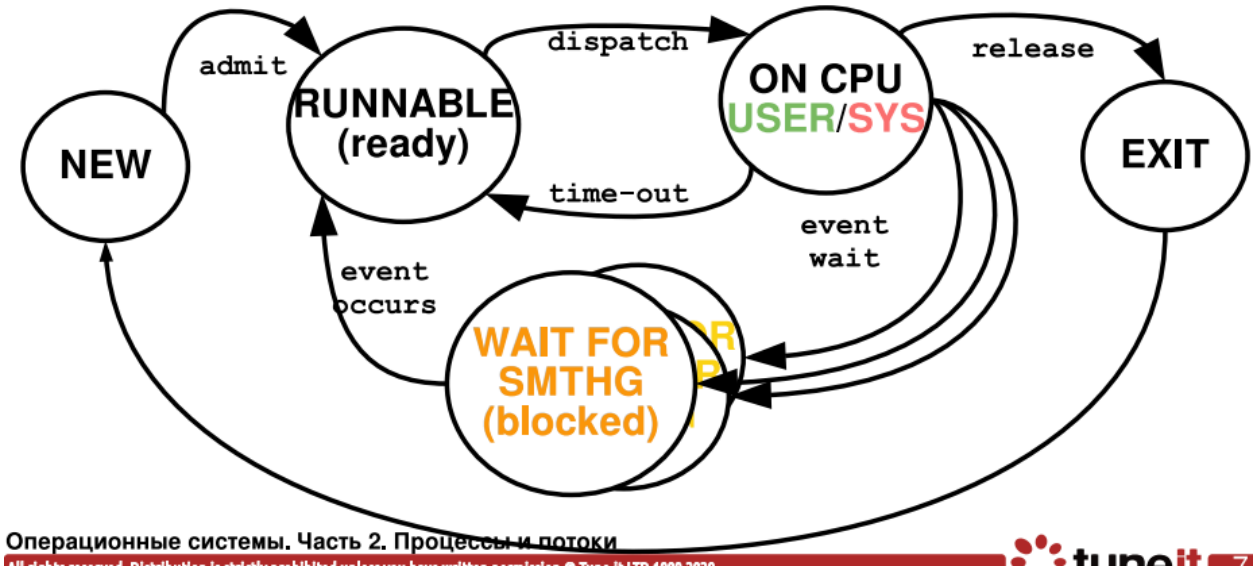
21. Сервисы, функции и важные компоненты Windows.

Функции Windows API (CreateProcess, CreateFile)
Syscalls (NtCreateUserProcess)
Функции Kernel Support (ExAllocatePoolWithTag)
Windows services
Dynamic Link Libraries (DLL)
Hyper-V гипервизор
Firmware, Registry, Объекты и безопасность, оснастки

22. Процесс, характеристики процесса в момент выполнения. Состояние процесса.

Процесс - выполняемая программа / экземпляр программы выполняемый на компьютере / единица активности зарактеризуемая выполнением последовательных команд темкущим состоянием и связанным с ней множеством системных ресурсов.
Характеристики процесса: pid, состояние, приоритет, счетчик команд, указатель на область памяти процесса, контекст (регистры, user/kernel), статус io, счетчик системных ресурсов
Состояния: NEW, EXIT, RUNNABLE, RUNNING, WAIT

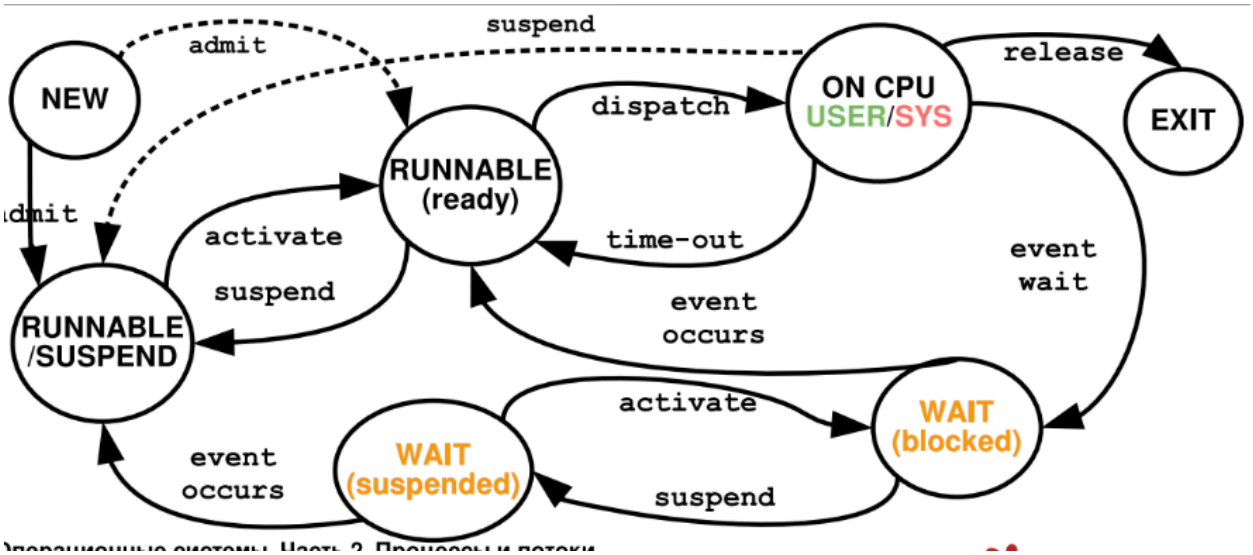
23. Модель процесса с пятью состояниями, назначение состояний.



Состояния: NEW, EXIT, RUNNABLE, RUNNING, WAIT
NEW - только созданный процесс, не загружен в основную память
RUNNABLE - готов к выполнению
WAIT - ожидает что что то завершится например IO
RUNNING - выполняется на процессоре
EXIT - процесс не может продолжить выполнение, структуры процессов все еще есть

24. Paging и Swapping. Модель процесса с семью состяниями.

Paging - выгрузка/загрузка неиспользуемых страниц процесса на диск
Swapping - выгрузка всего процесса кроме критически важных для ядра структур управления
Придумали потому что памяти всегда мало



NEW новый процесс
RUNNABLE готов к выполнению
ON CPU выполняется
RUNNABLE/SUSPEND готов к выполнению но частично в свопе
WAIT BLOCKED - ожидание если процесс долго блокируется, выгружен в своп
WAIT SUSPENDED остановлен и выгружен в своп
EXIT - завершен

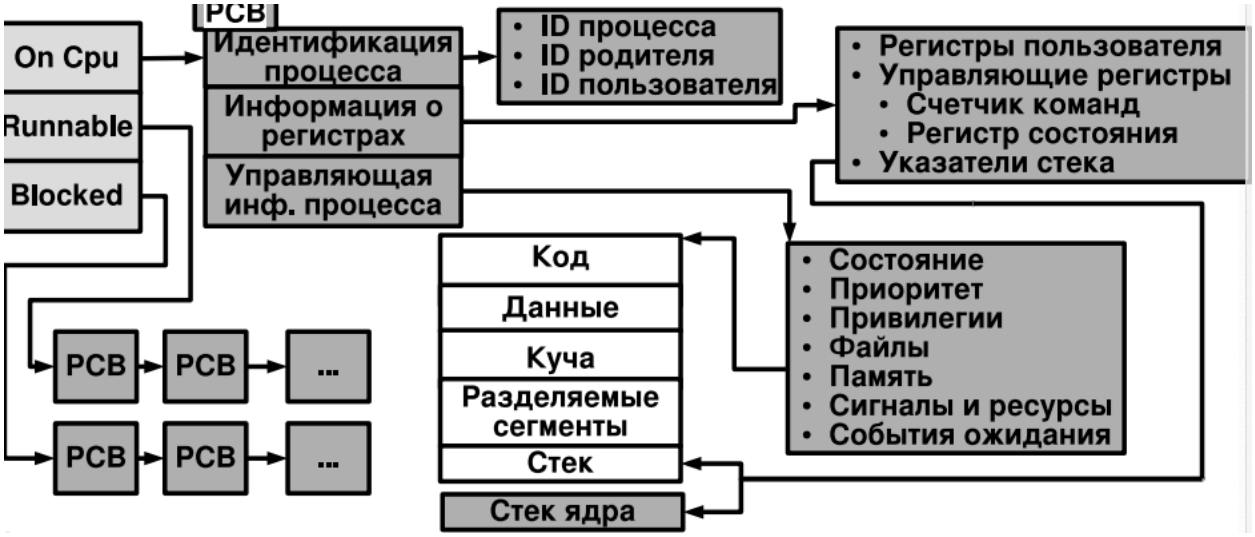
25. Управляющие таблицы процесса. Образ процесса.

ОС использует управляющие таблицы процесса для хранения информации о состояниях процессов.
Образ процесса - представление процесса в памяти
Существуют управляющие таблицы для: памяти, файлов, устройств, процессов
Таблица процессов хранит указатели на области памяти с образами процессов
Образ процесса включает в себя: pid, состояние, управляющая информация, стек, закрытое адресное пространство (программы и данные), открытое адресное пространство

26. Управляющий блок процесса (PCB), состав PCB.

PCB - структура данных для хранения всей информации о процессе, создается при инициализации процесса.
Состоит из:
Индентификации процесса (pid, id родители id пользователя),
информации о регистрах (управляющие регистры, указатели стека, регистры пользователя),

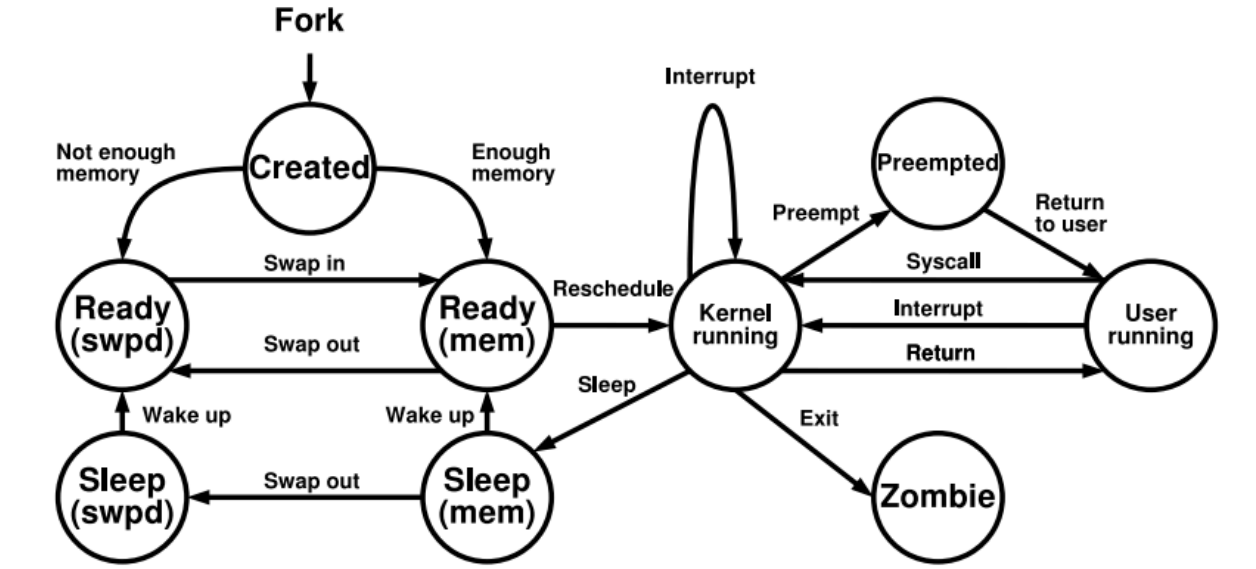
управляющая информация процесса (состояние,приоритет, и т д), память, код данные куча

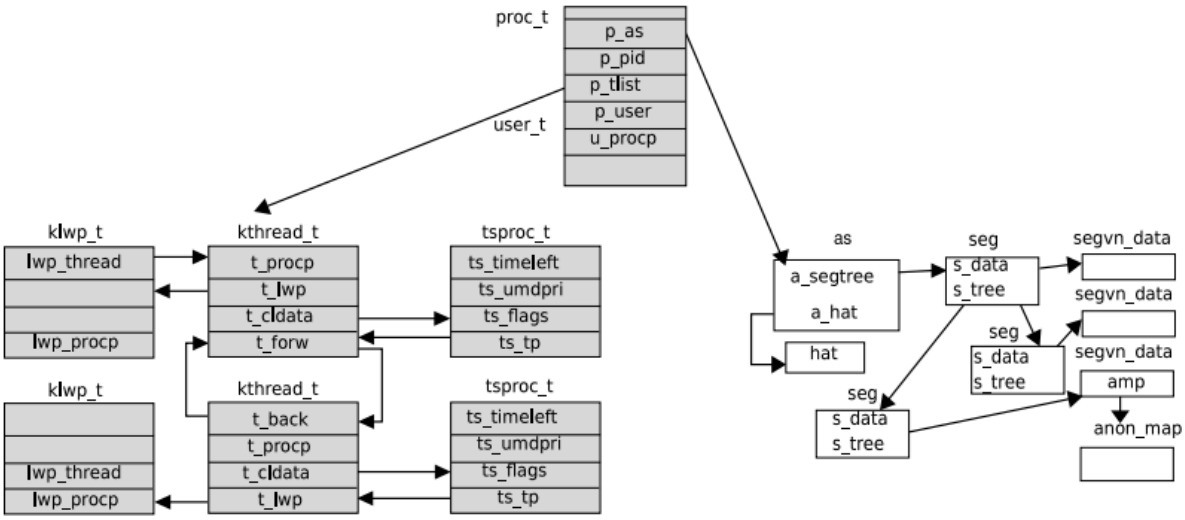


27. Функции ОС, связанные с процессами. Создание процесса, переключение процессов.

Управление процессами: создание/завершение, планирование/диспетчеризация, переключение, синхронизация и поддержка обмена информацией между процессами, организация PCB.
Управление памятью: выделение адресного пространства, пейджинг/свопинг, управление страницами и сегментами
Управление вводом-выводом: управление буферами, выделение процессам каналов io
Функции поддержки: обработка прерываний, контроль системы, учет ресурсов
Создание процесса: присвоит pid, выделить память, инициализировать PCB, поставить процесс в очереди ядра, создать потоки IO, создать другие управляющие структуры данных
Переключение процессов: для переключения между процессами используется механизм прерываний (io/trap/syscall), ОС сохраняет состояние в PCB.
Процессы могут работать в user/kernel mode, для перехода между режимами так же используется прерывание.

28. Процессы в ОС UNIX SVR4. Диаграмма состояний, основные структуры.



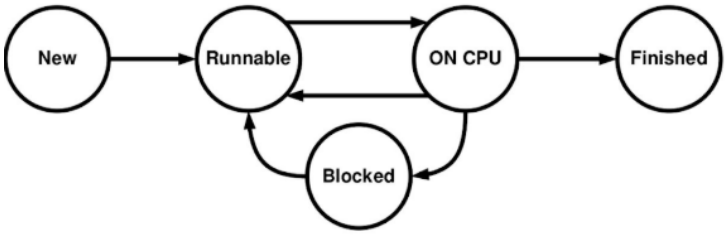


29. .Понятие потока выполнения, связь потока и процесса. Преимущества потоков.

Процесс - единица группировки общих ресурсов
Поток - единица выполнения программного кода
Поток содержит:
Состояние выполнения
Регистры

Стек (ядра и пользователя)
Локальные переменные
Доступ к памяти и другим ресурсам процесса владельца
Преимущество потоков:
Создаются переключаются завершаются быстрее
Модуллная структура программы

30. Состояния потока, User Level Threads vs Kernel Level Threads



ULT - реализуются библотеками на стороне полльзователя

KLT - реализуются ядром

Существуют разные соотношения 1:1, M:1, 1:M, M:N потоков

31. Многопроцессорность и многопоточность. Закон Амдала.

Многопроцессорность - управление несколькими процессорами в многопроцессорной системе
Закон Амдала: Ускорение = (работа на одном процессоре) = (работа на N процессорах) = 1/((1-f)+f/n) T - время работы, f - доля распараллеливания
В реальности f лежит в диапазоне от 0.2 до 0.8, в редких случаях 0.9
График показывает что при увеличении процессов даже с большой долей распараллеливания мы выходим на плато где можно задать вопрос целесообразности покупки кучи процессоров

32.Механизм параллельных вычислений, функции ОС.

Функции ОС:
Отслеживание ресурсов п/п
Распределение и освобождение ресурсов для каждого активного п/п
Защита ресурсов от других п/п
Независимость от скорости выполнения п/п

33. Проблемы параллельного выполнения: взаимоисключения, взаимоблокировки, голодание. Требования к взаимным исключениям. Уровни взаимодействия процессов и потоков.

Проблемы:
Взаимоисключения - процессы/потоки одновременно испольщуют критический ресурс
Взаимоблокировки - (dead/live locks) процессы/потоки одновременно захватывают требуемые ресурсы
Голодание - конкуренция за ресурсы порождает невозможность доступа к ресурсу
Требования: взаимоисключения принудительны, процесс/поток не должен влиять на другие п/п в некритическом случае, противодействие бесконечному ожиданию доступа к критическому участку, ограничение времени нахождения в критических участках, незамедлительный вход в свободный критический участок.
Уровни взаимодействия: процессы/потоки ничего не знают друг о друге
п/п сотрудничают используя общий ресурс (взаимоисключения, взаимоблокировки, голодание, связь данных)
п/п совместно выполняются (взаимоблокировки голодание)

34. Примитивы синхронизации ОС. Предназначение примитивов синхронизации

Базовые механизмы ОС для обеспечения корректной работы между процессами и потоками которые работают с общими ресурсами. Задача: избежать гонок данных, взаимных блокировок и тд
Примитивы: Семафоры (захват или освобождение множественного ресурса), бинарные семафоны (для одного ресурса), Мьютекс (блокировка/освобождение ресурса единственным потоком), условные переменные (блокировка до выполнения какого-либо условия), блокировки чтения/записи, монитор, флаги событий, почтовые ящики

35. Примитивы синхронизации ОС. Семафоры и мьютексы. Бинарный семафор

Семафор - структура данных в ядре которая управляет состоянием ресурса
Три операции с семафорами - инициализация/инкремен/декремент (атомарные)
Счетчик семафора управляет доступом к ресурам, используется для множественных ресурсов

Бинарный семафор - как обычный но принимает только значения 0/1, используется для одного ресурса

Mutex - механизм гарантирующий что только один поток или процесс может одновременно использовать общий ресурс.
Тот кто захватил блокировку тот и должен ее освободить
Существует несколько типов мьютексов например: Блокирующий (если поток не может захватить мьютекс то он блокируется), Спин (активное ожидание при невозможности заблокировать), адаптивный (сначала спин потом блокируется)

36.Примитивы синхронизации ОС. Условные переменные, rwlocks.

Условные переменные позволяют потоку ждать выполнения определенного условия прежде чем продолжить выполнение.
3 метода для работы с условными переменными
Wait - блокирует поток
Signal - пробуждает один ожидающий поток
Broadcast - пробуждает все потоки

RWLocks - механизмы синхронизации которые позволяют нескольким потокам читать но одному писать
Когда читатели читают они захватывают readlock, в нем отображается кол-во одновременных читателей
Когда надо читать - устанавливаем требования записи и ждем
Rwlock ждем освобождения readlock, оповещает писателя, оповещает читателей

37. Примитивы синхронизации ОС. Мониторы, флаги событий, передача сообщений.

Мониторы - высокоуровневый механизм синхронизации, объединяет: объект/ресурс, мьютекс, условные переменные для ожидания и уведомления потоков
Программисту не требуется возиться с захватом освобождением или ожиданием и нотификацией, в ОС не используется

Event Flags - биты указывающие на выполнение определенного события
Используются для синхронизации потоков когда один поток ждет выполнение другого
Операции с флагами: установка, сброс, ожидание, ожидание всех флагов

Message Passing - механизм обмена данными между процессами или потоками.
Процессы обмениваются через очереди сообщений, каналы, сокеты
Две операции - получить/отправить
Прямая адресация, косвенная адресация
Синхронизация посылки и получения: Блокирующая, неблокирующая, ганантия доставки
Формат: фикс/переменная длинна/файл
Выборка из очереди: фифо/приоритет

38. Примитивы синхронизации ОС. Неблокирующие примитивы синхронизации и неблокирующие структуры данных.

Блокировка вызывает переключение контекста а это дорого

Wait free - метод свободен от ожидания если каждый вызов завершается за конечное число шагов вне зависимости от других потоков
Lock free - метод свободен от блокировок если хотя бы один из вызовов метода продвигается вперед (имеет глобальный прогресс) вне зависимости от других потоков
Obstruction free - если он завершается за конечное число шагов если конкурирующие вызовы не исполняются
Неблокирующие структуры данных - списки деревья очереди

39. Управление памятью, основные определения и требования к организации.

Основная память - где процессор может исполнять программы
Вторичная память - где программы и данные могут храниться в том числе во время выполнения
Кадр - область фикс размера в основной памяти
Страница - область фикс размера во вторичной памяти которая может быть скопирована кадр
Сегмент - область переменного размера которая может быть разделена на страницы

Требования: переместимост, защита, совместное использование, логическая организация, логическая организация (управление основной и вторичной памятью, модульная организация), физическая организация (быстрый доступ к основной, вторичная дороже но обеспечивает долговременное хранение)

40. Фиксированное и динамическое размещение программ в памяти.

Фиксированное размещение - программы делятся на кадры фиксированного размера
Из минусов: неэффективно для маленьких процессов, сложности если не помещается в раздел, ограничение на кол-во одновременных процессов

Динамическое размешение - блоки разного размера
Минусы: фрагментация, сложные и медленные алгоритмы размещения в памяти
Подходит для загрузки драйверов, компромисс - buddy system

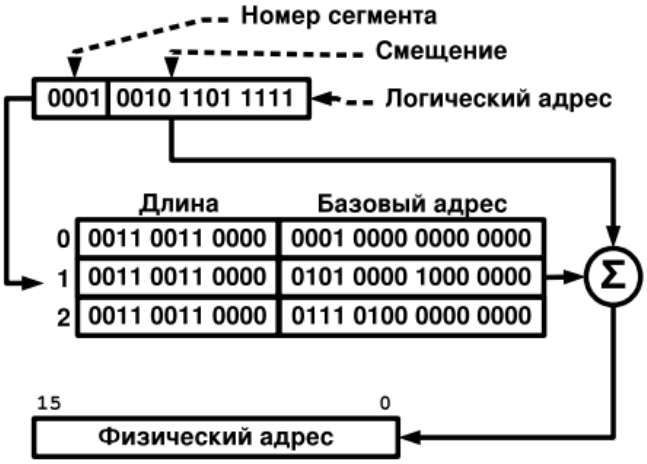
41. Модели аппаратного перемещения программ.

Копировать програаамы средствами ОС очень дорого

- Модели аппаратного перемещения:

42. Простой страничный поход и простая сегментная организация.

Страничный подход: делим память на кадры небольшого одинакового размера
Страница занимает один кадр, размер кратен 2 для удобства
Уменьшается внутренняя фрагментация, внешняя исчезает



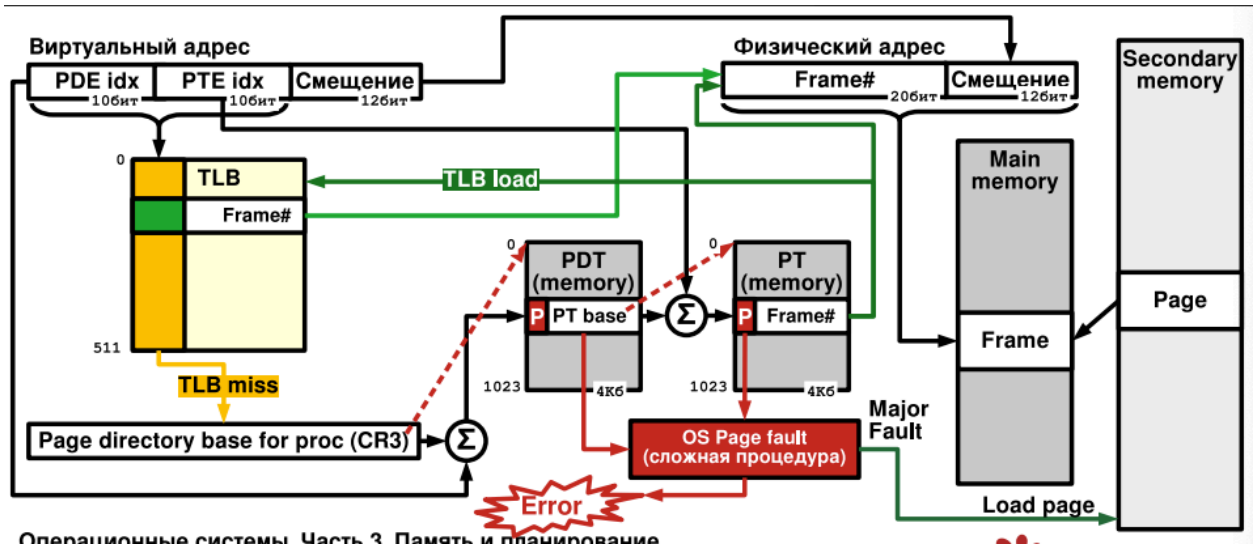
Необходима таблицы страниц
Таблица сегментов
Программист устанавливает длину сегмента и базовый адрес

43. Виртуальная память основные определения и принципы организации аппаратуры и управляющих программ.

Физический адрес - адрес в основной памяти
Виртуальный адрес - адрес внутри процесса

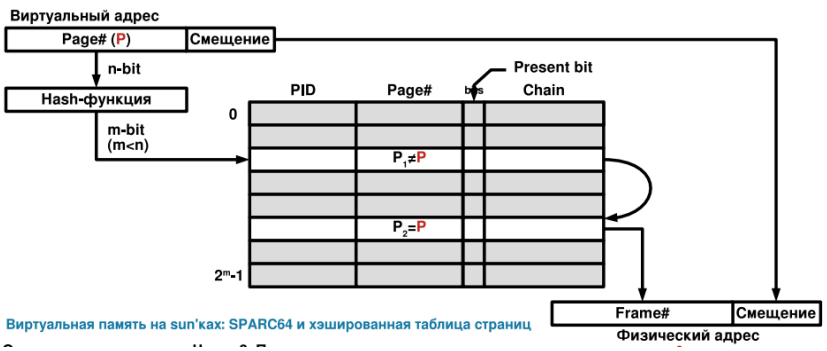
Адресное пространство - диапазон адресов процесса
Виртуальное адресное пространство - область для одного процесса с виртуальными адресами
Виртуальная память - схема расположения процессов в памяти (вирт адреса транслируются в основную память, размер памяти ограничен схемой адресации, вторичная память адресуется так же как и основная) совокупность аппаратных и программных средств
Resident set - часть процесса в основной памяти
Real memory - память где процесс выполняется
Virtual memory - память которую процесс может занять
Разные структуры организации: одноуровневая многоуровневая инвертированная

44. Виртуальный страничный обмен. Двухуровневая организация MMU и TLB 80386. (для КОТ и ГТ — общие принципы)



Виртуальный адрес → TLB → Каталог страниц → Таблица страниц → Физический адрес.

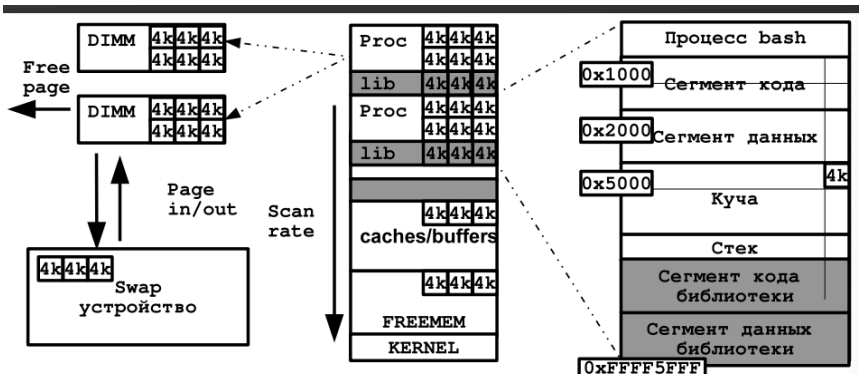
45. Инвертированная таблица страниц.



Метод управление виртуальной памятью, уменьшает объем необходимой для хранения таблиц страниц памяти
Преобразует виртуальный адрес в физический.

Виртуальный адрес: page# + смещение
Записи в таблице: pid + page# + бит наличия в памяти + chain (указатель для обработки коллизий хеш функции (следующий элемент))/frame#
Физический адрес: frame# + смещение

46. Сегментно-страничная виртуальная память.



У каждого процесса есть таблица сегментов, в сегменте таблица страниц
Сегментный адрес используется для нахождения базового адреса сегмента через таблицу сегментов
Виртуальный адрес внутри сегмента преобразуется в номер страницы и смещения, транслируется в физический адрес
Комбинирует преимущества сегментной и страничной памяти (гибкость, удобство управления)

47. Влияние размера страницы виртуальной памяти на ОС. Стратегии ОС по работе с виртуальной памятью.

При увеличении размера страницы: меньше записей в таблице,меньше page fault, уменьшает кол-во промахов tlb, больше фрагментация, сокращают кол-во io операций
При уменьшении размера страницы: больше записей в таблице, больше page fault, больше промахов tlb, меньше фрагментация, меньше локальности данных
Intel 4kb, 2mb, 1gb

Работа ОС с вирт памятью:
Выборка из вторичной памяти: по требования / предварительная
Стратегия размешения: рядом с процессором работающим с этими данными для NUMA
Стратегия очистки: по требования / предварительная
При завершении процесса все страницы выгружаются
ОС может выгружать резидентную часть процессов (swapping)
Стратегии замещения: оптимальная lru fifo, clock

48. Стратегии замещения страниц ОС. Часовой Алгоритм. Управление резидентной частью процесса.

Нужно понимать какие страницы загружать и выгружать из памяти и желательно это предсказывать. Можно учитывать прошлое поведение.
Некоторые фреймы нельзя выгружать (части ядра, буферы), а некоторые используются очень часто и их выгрузка повлияет на процессы
Стратегии: оптимальная стратегия (невозможная стратегия: мы условно знаем что нам понадобиться а что нет)
LRU (заменяем наименее используемые страницы (считаем кол-во использований))
FIFO (просто но не эффективно)
Clock: для каждой страницы бит изменения, если обращаемся то ставим бит. Одна стрелка сбрасывает биты изменения, другая ставит если обращались
Страницы без биты вытесняем

Управление резидентной частью процесса: нет смысла держать все страницы, после N-страниц выделения памяти не снижает кардинально кол-во page faults.
Стратегии: динамический и фикс размер.

49. Виды планирования процессов. Критерии краткосрочного планирования. Приоритеты.

Долгосрочное (о добавлении в пул выполняемых процессов)
Среднесрочное (о добавлении полностью/частично в основную память)
Краткосрочное (о том кто будет выполняться процессором)
Планирование IO
Критерии краткосрочного планирования:
Пользовательские: время оборота, время отклика, предельное время,предсказуемость
Системные: пропускная способность, загруженность процессора, справедливость, принудительная приоритизация, сбалансированная загрузка
Приоритеты: цифровой приоритете процессам, выбирается наивысший, голодание низких приоритетов, могут изменяться динамически, при совпадении доп стратегии

50. Использование приоритетов.

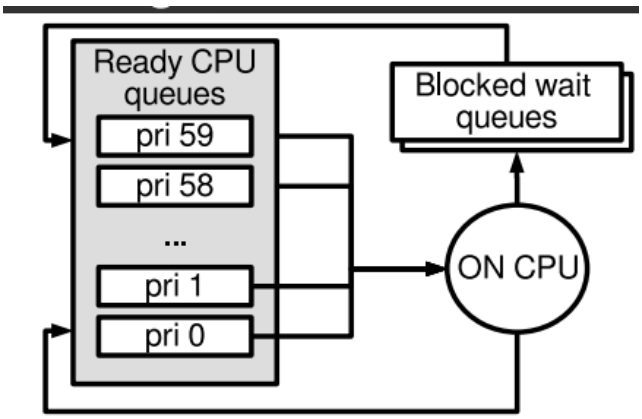
Приоритеты: цифровой приоритете процессам, выбирается наивысший, голодание низких приоритетов, могут изменяться динамически, при совпадении доп стратегии

51. Стратегии планирования FCFS, RR, SPN, SRT, HRRN, Feedback.

FCFS - аналог FIFO

RoundRobin - каждому процессу дается фиксированное время выполнения, если не завершил - в конец очереди
Shortest Process Next - выбираются самые быстрые процессы (голодание больших)
Shortest Remaining Time - впереди идут те процессы которым меньше всего осталось выполнятся
Highest Response Ration Next - приоритеты расставляются как (время ожидания + выполнения) / время выполнения
Feedback - приоритет снижается если он слишком долго или часто блокируются

52. Feedback планировщик и классы планирования ОС UNIX SVR4.



Feedback - приоритет снижается если он слишком долго или часто блокируются. Обеспечивается баланс между короткими и длинными процессами. Если процесс не успевает выполнится за квант времени то в конец очереди
Globpri - глобальный приоритет
Quantum - тайм квант
Tqexp - время за которое процесс может выполнится на cpu
Tqexp - приоритет после кванта времени
Slpret - приоритет после ожидания
Maxwait - максимум ожидания перед повышением приоритета, lwait приоритет который ждал максимум времени
Классы планирования:
TimeSharing - разделение времени (feedback)
InterActive - буст к активному приложению
Fixed, System, RealTime - фиксированный приоритет
FairShare - справедливый планировщик
Отдельные приоритеты для потоков прерывания
Учет близость памяти для NUMA

53. Справедливое планирование.

Процессы делятся на классы, каждому классу выделяется shares.
Каждой группе процессов выделяются кванты администрирования, доля ресурсов - отношение shares к их суммме.
Учитывает все процессоры в системе

54. Планирование в многопроцессорных системах. Типы многопроцессорных систем с точки зрения организации планирования. Гранулярность и проектирование планировщиков процессов и потоков для многопроцессорных систем.

Типы систем: Слабосвязанные (своя память, ввод-вывод), функционально-специализированные(master cpu, slave cpu's), сильносвязанные процессоры (общая память и кеш, одна ос)
Гранулярность: (частота синхронизации)
Fine (<20 команд), Medium (20-200), Coarse(200-2000), VeryCoarse(2000-1m), Independent
Проектирование планировщиков процессов:
Назначение процессов процессорам (статическое/динамическое/балансировка)
Влияние выбора алгоритма диспетчиризации снижается
Подходы:
Loadsharing (равномерное распределение нагрузки, нет центрального планировщика, блокировки центральной очереди, неэффективность при тонкой и средней гранулярности)
Gang Scheduling - связанные потоки на связанные процессы по одному на процессор
Dedicated processor assignment - пул процессоров равный кол-ву потоков
Динамическое планирование - при свободных cpu забирают их из процесса с несколькими cpu

55. ОС реального времени и планировщики. Deadline-планирование.

ОС реального времени для выполнения задач с жесткими ограничениями по времени (оборудование авиация и тд)
Hard realtme - все задачи должны быть выполнены в жестко заданное время
Soft realtime - небольшие задержки допустимы
Основные требования:
Determinism - гарантия выполнения операции в заданное время
Responsiveness - сколько требуется системе на реакцию на событие
User Control - пользователь может задать приоритеты и управлять выполнением задач
Reliability - выше требования к надежности
Fail-soft operation - мягкая реакция на ошибки
Планировщик: строгие приоритеты, минимальные задержки, вытеснение задач

Deadline планирование:
Подход для задач которые должы быть выполнены в строго заданное время
Своевременное завершение критически важно, конфликты и нехватка ресурсов должны быть устранены

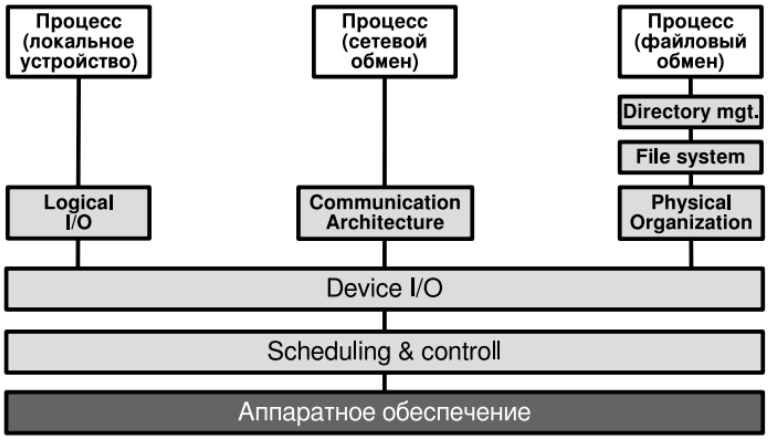
Ready time, Starting deadline, completion deadline, processing time, resource requirements, priority, subtask structure

56. .Проблема инверсии приоритетов, типы инверсии и способы решения в планировщике.

Инверсия приоритетов - это ситуация, когда процесс с высоким приоритетом не может выполняться, потому что ожидает ресурса, удерживаемого процессом с низким приоритетом, в то время как процесс со средним приоритетом занимает процессор.
Типы: ограниченная (ограничевается временем необходимым для завершения работы процесса с низким приоритетом) неограниченная (может длиться бесконечно), (цепочка блокировок - несколько процессов создают цепочку зависимостей где все ждут освобождения ресурса другими)
Решение: наследование приоритетов, задача с низким приоритетом наследует приоритет любой высокоприоритетной задачи ожидающей совместный ресурс.

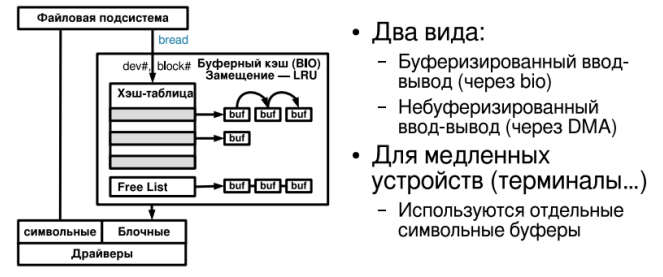
57. Ввод-вывод. Современные устройства и скорости обмена, развитие способов ввода- вывода, логическая структура ввода-вывода.

Устройства: пользовательские, внутренние, связи
Характеристики: скорость, применение,сложность управления, единица передачи, представление данных, обстоятельства ошибок.

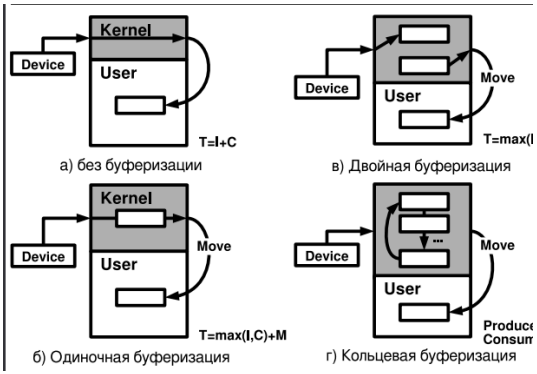


Развитие: программируемый ввод-вывод (процессор управляет устройством через его шину/контроллер подключается к шине и имеет набор управляющих регистров), ввод-вывод с использованием прерываний, прямой доступ к памяти (Direct Memory Access) перенос буфера контроллера в область памяти (контроллер - отдельный вычслительный модуль)

58. Буферизация ввода вывода. Ввод-вывод в UNIX SVR4.



- Два вида:
  - Буферизированный ввод-вывод (через bio)
  - Небуферизированный ввод-вывод (через DMA)
- Для медленных устройств (терминалы...)
  - Используются отдельные символьные буферы



59. Диски и дисковое планирование.

Hard drive - блины и считывающая головка, интерфейсы ide, sata, sas, scsi
SSD - адресуемая память - интерфейсы sata, sas, nvme
Доступ к дисковому устройству: ожидание в очереди, ожидание канала, поиск, передача данных
Дисковое планирование:
FIFO, PRI (по приоритету процесса), LIFO (последний запрос первый), SSTF(ближайший запрос к текущему положению головки), SCAN(ездим туда сюда по диску и обслуживаем запросы), CSAN (ездим в одном направлении и возвращаемся в начало) N-step-SCAN (разбивается на подочереди и обрабатывается отдельно), FCSCAN - две одоочереди, обрабатывается одна пока вторая не заполнится

60.Концепции RAID.

Несколько дисков которые ОС логически объединяет в один
Данные распределены по всем дискам, избыточная информация, один блок на нескольких дисках, запросы могут быть выполнены параллельно
RAID 0, 1, 2,3,4,5,6 - различные конфигурации отличаются по скорость/отказоустойчивость

61. RAID-0, 1, 10, 0+1.

RAID 0: данные разбиваются на блоки и записываются на все диски одновременно, быстрая запись и чтение, нет избыточности, объем кол-во дисков * объем
RAID 1: mirroring: все данные дублируются, быстрое чтение, большие траты на хранение
RAID 10: смесь RAID 1, 0: сначала зеркалируются RAID1, потом разбиваются на блоки RAID0, высокая скорость записи и чтения, высокая надежность, минимум 4 диска, объем = половина общего объема
RAID 0+1: сначала разбивается на блоки RAID0, потом зеркалируются RAID1: высокая скорость, минимум 4 диска, объем = половина, меньше надежности чем в raid 10

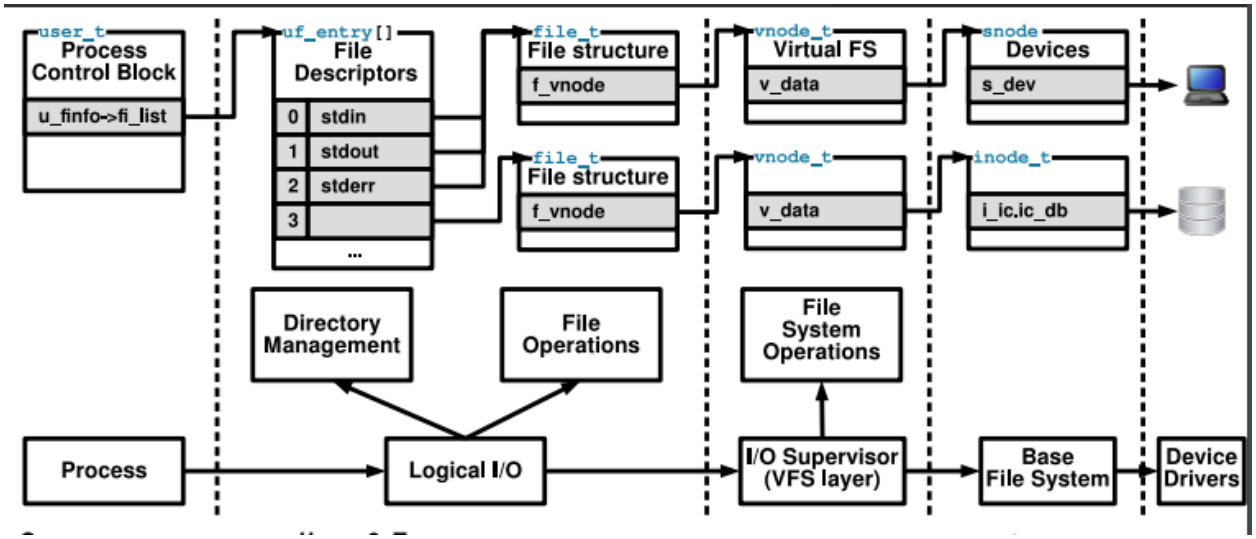
62. RAID 4,5,6. Аппаратные дисковые массивы.

RAID 4: striping, один диск для хранения четности (восстановления в случае сбоя), быстрая скорость чтения, объем n-1, диск четности узкое место
RAID 5: четность распределяется по всем дискам из массива, высокая скорость чтения, нет узкого места, возможность восстановления, объем n-1
RAID 6: в отличии от RAID5 хранит две копии честности, высокая надежность, высокая скорость чтения, объем n-2
Аппаратные дисковые массивы: сложные конструкции из RAID массивов, доп контроллеры для расчета четности, буфер записи, резервные батареи. Очень дорого быстро и отказоустойчиво

63. Файловый ввод-вывод, основные определения. Задачи ОС по управлению файлами. Совместное использование файлов.

Поле - одиночный элемент записи
Запись - набор полей
Файл - совокупность записей относящихся к однородному набору данных
База данных - файл со сложной взаимозависимой структурой полей и записей
Операции проводимые с файлом могут влиять на его структуру
Файл может не иметь структуры
Основные операции - создание удаления открытие закрытие чтение запись выбор позиции контрольные операции блокирование
Долгосрочное существование, совместное использование процессами
Задачи ОС: хранить, выполнять основные операции, гарантия корректности данных, перемещение, управление доступом, работа с файлами по именам удобная пользователю, минимизация потерь данных.
Совместное использование: файл не менялся процессами/потоками одновременно, использовал блокировки, должен иметь владельца, права доступа.

64. Управление файлами в UNIX SVR4



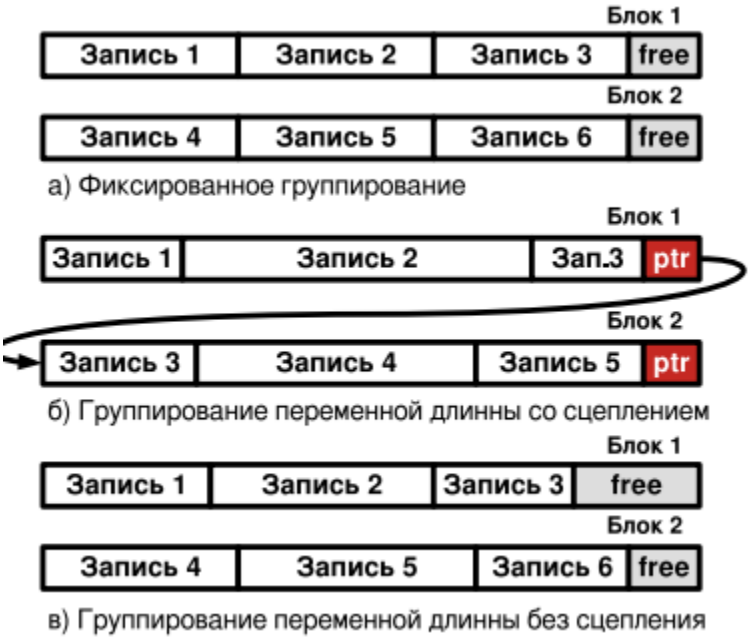
65.Каталоги файлов. Элементы каталога, операции ОС.

Каталоги - структура которая позволяет организовывать хранить и управлять файлами
Основные данные в структуре: имя файла, тип файла, организация файла,
Адресная информация: том, начальный адрес, размер
Доступ: владелец, права
Использование: создатель / дата создания, дата редактирования и тд

Каталог может быть полностью или частично загружен в основную память
Основные операции: поиск записей, создание/удаление записей о файле, получение списка записей, обновление каталога, изменение записи
Как правило структура каталогов - дерево /связанный граф

66. Размещение записей и файлов в блоках данных. Сложность и типы организации размещения.

Запись - логическая единица доступа к структурированному файлу.
Блок - минимальаня физическая единица ввода-вывода
Чем больше блок - тем меньше операций IO
Увеличение фрагментации в блоке



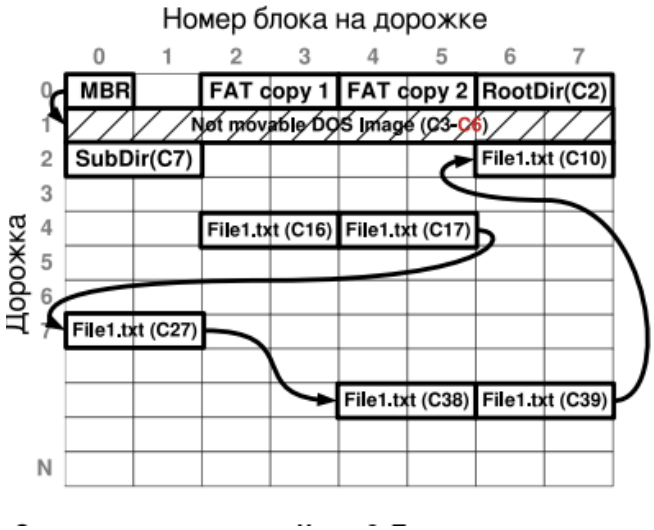
Типы группировки: фиксированное группирование, переменной длины со сцеплением, переменной длины без сцепления

Размещение файлов - способ хранения файлов на физическом диске. Файл состоит из блоков которые могут быть размещены по разному:
Непрерывное размещение, связанное размещение, индексное размещение
Проблемы размещения: фрагментация (внутренняя, внешняя), частота выделения блоков, скорость доступа, размер служебной информации о размещении файлов

67. Непрерывное размещение файлов (на примере ОС RT-11)

- Особенности
  - Предварительное выделение пространства для всего файла.
  - Один уровень каталога
  - Размер блока 512 байт
- Характеристики:
  - Внутренняя фрагментация мала
    Внешняя может быть большой — необходимо обязательное сжатие ФС
  - Выделение блоков для файла производится однократно
  - Невысокое время выделения блоков — если ФС пуста, то минимальное, если заполнена, то возможны ошибки размещения
  - Минимальный размер служебной информации о размещении файлов

68. Цепочечное размещение файлов (на примере DOS FAT)

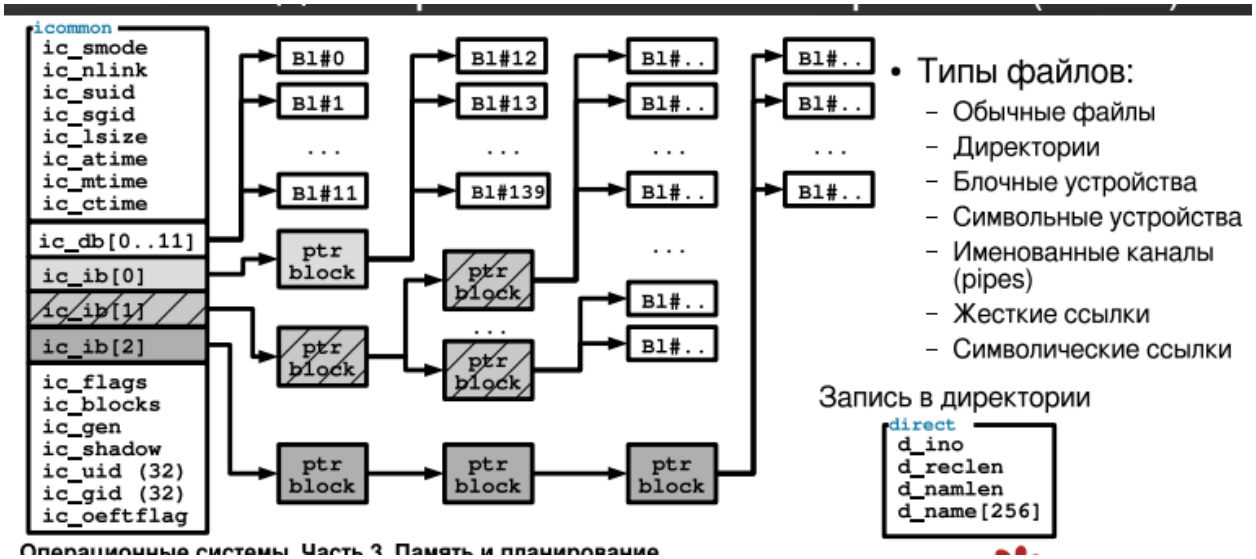


- Особенности
  - Выделение пространства для файла по мере необходимости
  - Размер порции файла — кластер (C) — от 512 (FAT12) до 32Кб (FAT 32)
  - Запись каталога содержит имя (8+3), размер файла, ..., номер начального кластера: File1.txt
  - FAT (File Allocation Table) — Содержит таблицу аллокации цепочки кластеров

CL#	10		16	17	27		38		39
Next Cl#		FFFF		17	27		38		39	10

- Характеристики:
  - Внутренняя фрагментация мала
  - Внешняя может быть большой — файлы размещены непоследовательно, необходимо обязательное дефрагментирование ФС для увеличения скорости
  - Высокое время выделения блоков файла
  - Размер служебной информации о размещении файлов ограничен FAT

69. Индексированное размещение (на примере файловой системы UNIX UFS)



- Типы файлов:
  - Обычные файлы
  - Директории
  - Блочные устройства
  - Символьные устройства
  - Именованные каналы (pipes)
  - Жесткие ссылки
  - Символические ссылки

Запись в директории

```
direct
d_ino
d_reclen
d_namlen
d_name[256]
```