1) **Первое поколение компьютеров** - электронно-вакуумные лампы, 1940-1950. Зарождение класса сервисных, управляющих программ (чтение и загрузка в оперативную память программ с внешних устройств). Зарождение языков программирования. Однопользовательский, персональный режим. Первый язык высокого уровня - FORTRAN, ЭВМ ENIAC.
2) **Второе поколение компьютеров** - полупроводниковые приборы: диоды и транзисторы, 1950-1960, пакетная обработка заданий, мультипрограммирование (одновременная обработка нескольких программ), аппарат прерываний, языки управления заданиями, первые прообразы файловых систем => виртуальные устройства, операционные системы (связано с этим поколением), БЭСМ-6 (Лебедев).
3) **Третье поколение компьютеров** - интегральные схемы малой и средней интеграции, 1960-1970, аппаратная унификация узлов и устройств, создание семейств компьютеров (программная преемственность), унификация компонентов программного обеспечения, IBM-360, PDP-11, драйверы устройств в ОС, UNIX, виртуальные устройства.
4) **Компьютеры четвертого поколения** - большие и сверхбольшие интегральные схемы, 1970-х – настоящее время. «Дружественность» пользовательских интерфейсов, сетевые технологии, безопасность хранения и передачи данных, Internet.
5) **Вычислительная система** - совокупность аппаратных и программных средств, функционирующих в единой системе и предназначенных для решения определенного класса задач.
6) **Структура вычислительной системы снизу-вверх:** аппаратные средства, управление физическими устройствами, управление логическими/виртуальными устройствами, системы программирования, прикладные системы.
7) **Физические ресурсы (устройства)** - компоненты аппаратуры компьютера, используемые на программных уровнях ВС или оказывающие влияние на функционирование всей ВС.
8) **Аппаратный уровень** - совокупность физических ресурсов вычислительной системы.
9) **Характеристики физических ресурсов:** правила программного использования (ключевое), производительность и/или ёмкость, степень занятости или используемости.
10) **Средства программирования, доступные на аппаратном уровне:** система команд компьютера, аппаратные интерфейсы программного взаимодействия с физическими ресурсами.
11) **Драйвер физического устройства** – программа, основанная на использовании команд управления конкретного физического устройства и предназначенная для организации работы с данным устройством.
12) **Уровень управления физическими устройствами** - совокупность драйверов физических устройств.
13) **Логическое/виртуальное устройство (ресурс)** – устройство/ресурс, некоторые эксплуатационные характеристики которого (возможно все) реализованы программным образом.
14) **Драйвер логического/виртуального ресурса** - программа, обеспечивающая существование и использование соответствующего ресурса.

15) **Ресурсы вычислительной системы** - совокупность всех физических и виртуальных ресурсов.
16) **Операционная система** - это комплекс программ, обеспечивающий управление ресурсами вычислительной системы.
17) **Система программирование** – это комплекс программ, обеспечивающий поддержание жизненного цикла программы в вычислительной системе.
18) **Жизненный цикл программы в ВС:** проектирование, кодирование или программирование, тестирование и отладка, ввод программной системы в эксплуатацию (внедрение) и сопровождение.
19) **Проектирование** - исследование задачи, исследование характеристик объектной среды (как объектная среда будет связана с нашей системой) и инструментальной среды.
20) **Объектная среда** – это та ВС, в рамках которой продукт будет функционировать.
21) **Инструментальная среда** – это ВС, которая будет использована для разработки программ.
22) **Кодирование** - построение кода на основании спецификаций при использовании языков программирования, трансляторов, средств для использования библиотек и средств для разработки программных продуктов. Результатом этапа кодирования являются исполняемые модули, объектные модули, исходные тексты программ и библиотеки.
23) **Тестирование** – это проверка спецификаций функционирования программы на некоторых наборах входных данных.
24) **Отладка** – процесс поиска, анализа и исправления зафиксированных при тестировании и эксплуатации ошибок.
25) **Средства программирования, доступные на уровне системы программирования** - программные средства и компоненты системы программирования, обеспечивающие поддержание жизненного цикла программы.
26) **Прикладная система** – программная система, ориентированная на решение или автоматизацию решения задач из конкретной предметной области.
27) **Виртуальная машина** - совокупность всех средств, доступных программисту или пользователю для взаимодействия с ВС на различных уровнях доступа (аппаратном, физическом, логическом, систем программирования, прикладных систем).
28) **Принципы работы компьютера Фон Неймана:** принцип двоичного кодирования, принцип хранимой программы (все представляется единым образом в едином устройстве памяти), принцип программного управления (принцип последовательной обработки).
29) **Оперативно запоминающее устройство (ОЗУ)** – устройство хранения данных, в котором размещается исполняемая в данный момент программа.
30) ОЗУ состоит из **ячеек памяти** (устройство, в котором размещается информация), которые состоят из **тегов** (поле служебной информации) и машинных слов (поле программно изменяемой информации). Тег может быть разрядом четности, разрядом команда-данные, разрядом тип данных.
31) **Производительность оперативной памяти** - скорость доступа процессора к данным, размещенным в ОЗУ.

32) Время доступа (access time- t_{access}**)** - время между запросом на чтение слова из оперативной памяти и получением содержимого этого слова.

33) Длительность цикла памяти (cycle time - t_{cycle}**)** - минимальное время между началом текущего и последующего обращения к памяти. ($t_{cycle} > t_{access}$)

34) Расслоение ОЗУ – один из аппаратных путей решения проблемы дисбаланса в скорости доступа к данным, размещенным в ОЗУ и производительностью ЦП.

35) Центральный процессор - обеспечивает последовательное выполнение машинных команд, составляющих программу, размещенную в оперативной памяти. Осуществляется выбор машинного слова, содержащего очередную машинную команду, дешифрация команды, контроль корректности данных, определение исполнительных адресов операндов, получение значения операндов и исполнение машинной команды.

36) Регистровая память – совокупность устройств памяти ЦП (регистров), предназначенных для временного хранения операндов, информации, результатов операций. (Регистры общего назначения, Специальные регистры: счетчик команд - адрес очередной выполняемой команды, слово – состояние процессора - режимы работы процессора, значения кодов результата операций, регистр указатель стека)

37) Устройство управления – координирует выполнение команд программы процессором.

38) Арифметико-логическое устройство — выполнение команд, арифметическая или логическая обработка операндов.

39) Рабочий цикл процессора – последовательность действий, происходящая в процессоре во время выполнения программы.

40) КЭШ память - (Аппаратное решение) буферизация работы процессора с оперативной памятью. (нахождение данных - **попадание**, если искомых данных нет - **промах**).

41) При возникновении промаха происходит обновление содержимого КЭША - **вытеснение.**

42) Стратегии вытеснения КЭШ памяти — случайное, наименее популярного.

43) Процесс вытеснения — сквозное кэширование, кэширование с обратной связью.

44) Сквозное кэширование - при появлении команды записи менять содержимое соответствующего операнда в блоке КЭШа и в блоке оперативной памяти.

45) Кэширование с обратной связью - при появлении команд записи меняется содержимое машинного слова только в КЭШе, при вытеснении модифицированного блока его содержимое сбрасывается в ОП. (Идет работа с тегом модификации)

46) Прерывание - событие в компьютере, при возникновении которого в процессоре происходит предопределенная последовательность действий. (Короткое – не требует больших затрат. Фатальное – продолжение выполнения программы невозможно.)

47) Внутренние прерывания - инициируются схемами контроля работы процессора.

48) Внешние прерывания - события, возникающие в компьютере в результате взаимодействия ЦП с внешними устройствами.

49) Этап аппаратной обработки прерываний – завершение текущей команды, блокировка прерываний, сохранение (частичное) состояния процессора, программная обработка прерывания.

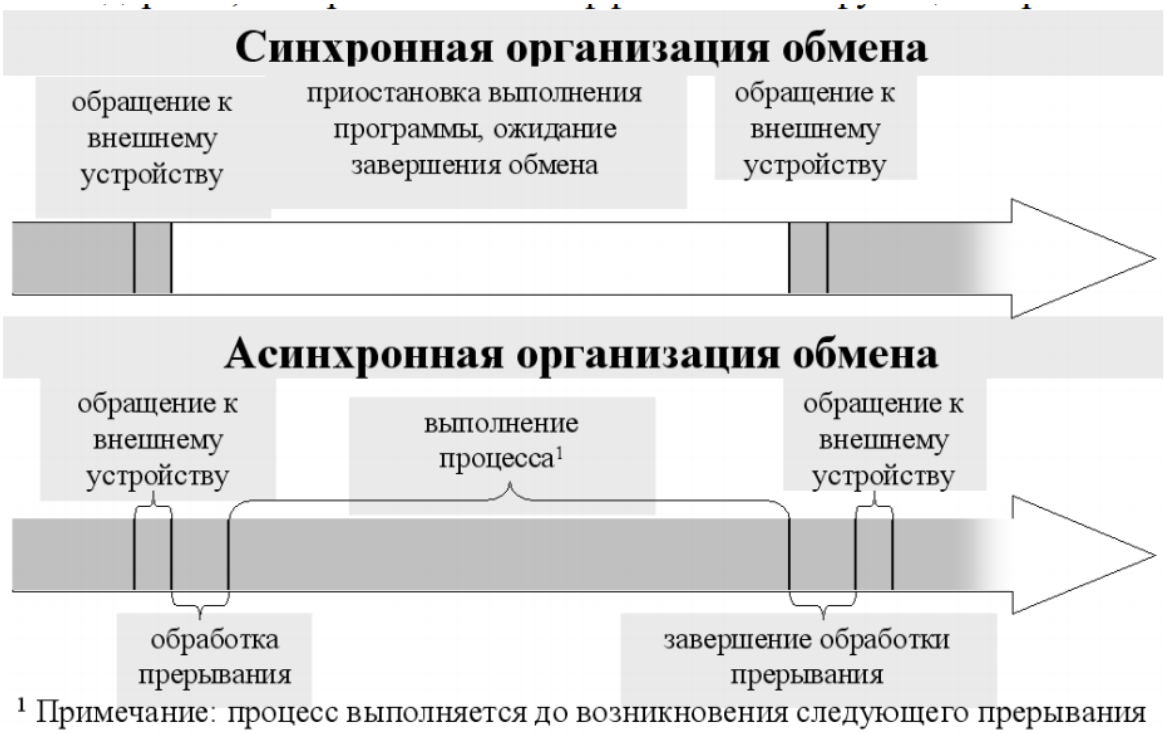
50) Программная обработка прерывания:

Рис. 33. Программный этап обработки прерываний.

51) ВЗУ Последовательного доступа: Магнитная лента.

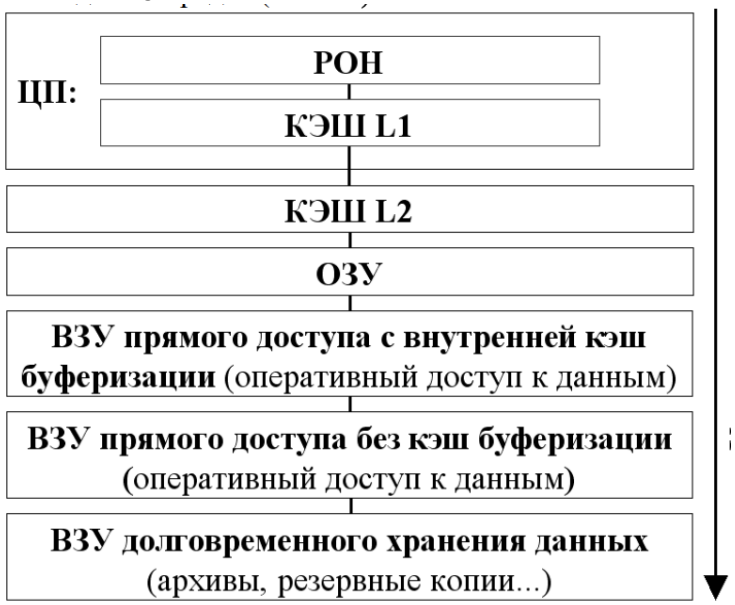
52) ВЗУ Прямого доступа: магнитные диски, магнитный барабан, магнито-электронные ВЗУ прямого доступа.

53) Модели синхронизации при обмене с внешними устройствами:



[1] Примечание: процесс выполняется до возникновения следующего прерывания

54) Иерархия устройств хранения информации:



55) Мультипрограммный режим - режим при котором возможна организация переключения выполнения с одной программы на другую.

56) Аппаратные средства компьютера, необходимые для поддержания мультипрограммного режима: аппарат прерываний (хотя бы прерывание по таймеру, чтобы зацикленная программа прерывалась и управление передавалось ОС), аппарат защиты памяти, специальный режим ОС (привилегированный - в этом режиме работы процессор может исполнить любую их своих машинных команд).

57) Аппарат виртуальной памяти – аппаратные средства компьютера, обеспечивающие преобразование программных адресов, используемых в программе адресам физической памяти в которой размещена программа при выполнении.

58) Базирование адресов – реализация одной из моделей аппарата виртуальной памяти. При базировании выделяется регистр базы, в котором будет храниться адрес, начиная с которого размещается программа. (решение проблемы перемещаемости программы по ОЗУ)

59) Страница - область адресного пространства фиксированного размера. Размер страницы - 2^K.

60) Страничная организация — аппаратная организация памяти, при которой все пространство делится на фрагменты одного размера (обычно 2^K).

61) Классификация Флинна - классификация архитектур многопроцессорных ассоциаций: поток управляющей информации – собственно команд (инструкций), и поток данных. Считаем потоки данных и команд независимыми (условно). Рассматриваем все возможные комбинации.

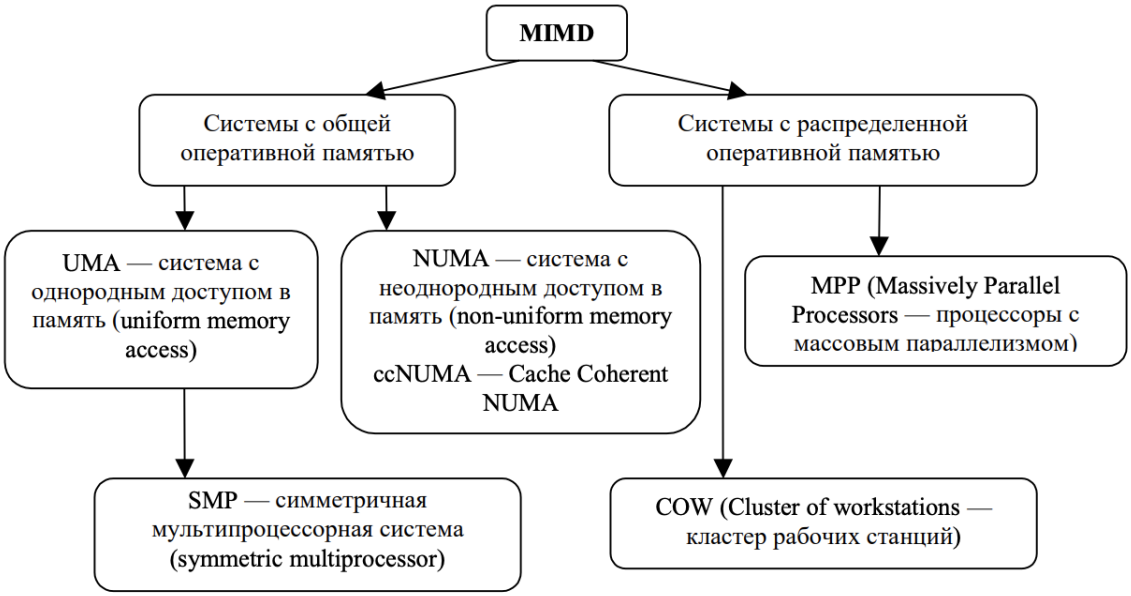
62) ОКОД (SISD – single instruction (одиночный поток команд), single data stream, (одиночный поток данных)) - традиционные компьютеры, которые мы называем однопроцессорными.

63) ОКМД (SIMD – single instruction(одиночный поток команд), multiple data stream(множественный поток данных)) - для каждой команды порция данных (векторная или матричная обработка данных)

64) **МКОД (MISD – multiple instruction(множественный поток команд), single data stream(одиночный поток данных))** – это вырожденная категория, считается, что ее нет. (Но есть вырожденные случаи — обработка графики или конвейерные системы)

65) **МКМД (MIMD - multiple instruction(множественный поток команд), multiple data stream(множественный поток данных))** — множество процессоров одновременно выполняют различные последовательности команд над своими данными.

66) **Иерархия MIMD-систем:**



67) В системе с **общей оперативной памятью** имеется ОЗУ и любой процессорный элемент имеет доступ к любой точке общего ОЗУ.

68) **UMA - Uniform Memory Access** — характеристики доступа любого процессорного элемента в любую точку ОЗУ не зависят от конкретного элемента и адреса (т.е. все процессоры равноценны относительно доступа к памяти).

69) **SMP:** (**+** простота реализации, **-** явная централизация - общая шина, ограничение на количество процессорных элементов, синхронизация кэша (решение - кэш-память с отслеживанием)

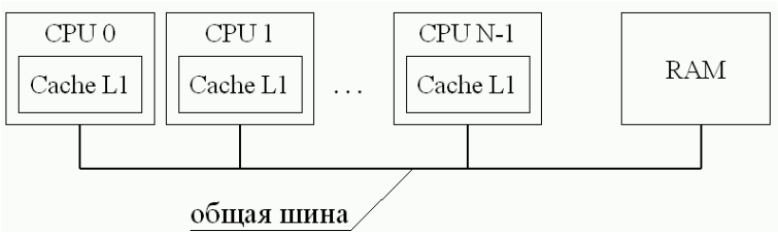


Рис. 60. Схема SMP-системы.

70) **NUMA - Non-Uniform Memory Access** - процессорные элементы работают на общем адресном пространстве, но характеристики доступа процессора к ОЗУ зависят от того, куда он обращается. Проблема с синхронизацией кэша осталась (можно юзать либо процессоры без кэша, либо юзать ccNUMA, но ccNUMA будет заполнять общую шину служебной информацией)

71) **Преимущества NUMA:** степень параллелизма выше, чем в SMP.

72) Системы с **распределенной оперативной памятью** представляются как объединение компьютерных узлов, каждый из которых состоит из процессора и ОЗУ, непосредственный доступ к которой имеет только “свой” процессорный элемент. Класс наиболее перспективных систем.

73) MPP – Massively Parallel Processors - Используются спец. средства коммуникации, более дорогие и более специализированные. Имеют высокую эффективность при решении определенного класса задач. Эти системы могут выстраиваться в различные топологии.

74) COW - Claster of Workstations - многомашинная система, машины в которой объединены специальной быстрой сетью. Преимущества: прозрачность архитектуры, относительная универсальность (большой круг задач). Минусы: топология (отвод тепла и коммуникации).

75) Терминальный комплекс – это многомашинная ассоциация предназначенная для организации массового доступа удаленных и локальных пользователей к ресурсам некоторой вычислительной системы. (Состав предполагает: основную ВС, локальные мультиплексоры, локальные терминалы, модемы, удаленные терминалы, удаленные мультиплексоры)

76) Виды каналов: коммутируемые (каждый раз новый), выделенные (на постоянной основе).

77) Количество участников общения: канал точка-точка (без мультиплексирования), многоточечный канал (через локальный мультиплексор).

78) Направление движения информации: симплексные каналы (в одном направлении), дуплексные (в двух направлениях), полудуплексные(в двух направлениях, но когда передает один, другой ждет).

79) Компьютерная сеть – объединение компьютеров (или вычислительных систем), взаимодействующих через коммуникационную среду. (свойства: большое число связанных узлов, распределение обработки информации, расширяемость сети, применение симметричных интерфейсов обмена информации внутри сети)

80) Коммуникационная среда – каналы и средства передачи данных.

81) Состав сети: абонентские машины (хосты) и коммуникационные (вспомогательные: шлюзы, маршрутизаторы).

82) Модели построения компьютерной сети: сеть коммутации каналов, сеть коммутации сообщений, сеть коммутации пакетов.

83) Сообщение - логически целостная порция данных, имеющая произвольный размер.

84) Сеанс связи состоит из обмена сообщениями между абонентами.

85) Сеть коммутации каналов - обеспечивает выделение коммуникаций абонентам на весь сеанс связи.

86) Сеть коммутации сообщений - взаимодействие представляется в виде последовательности обменов сообщений.

87) Сеть коммутации пакетов - все сообщения разделяются на блоки фиксированного размера - **пакеты** (структура пакеты: заголовок, данные).

88) **Модель ISO/OSI** – система открытых интерфейсов, состоит из 7 уровней: Физический (передача двоичной информации), Канальный (решаются задачи обеспечения передачи данных, вопросы синхронизации и доступности физической линии), Сетевой (решаются задачи взаимодействия сети: обеспечивается связь между взаимодействующими устройствами), Транспортный (программное взаимодействие: корректная транспортировка данных, может обеспечиваться выявление исправление ошибок при передаче), Сеансовый (управление сеансами связи: определение активной стороны, подтверждение полномочий и т.п.), Представительский (унификация кодировок и форматов данных), Прикладной (формализуются правила взаимодействия с прикладными системами). В каждом уровне модель ISO/OSI предполагает наличие некоторого количества протоколов, каждый из которых может осуществлять взаимодействие с одноименным протоколом на другой взаимодействующей машине (возможно виртуальной).

89) **Протокол** — формальное описание сообщений и правил, по которым сетевые устройства (вычислительные системы) осуществляют обмен информацией. Правила взаимодействия одноименных (одноранговых) уровней сети.

90) **Интерфейс** – правила взаимодействия вышестоящего уровня с нижестоящим.

91) **Служба или сервис** – набор операций, предоставляемых нижестоящим уровнем вышестоящему.

92) **Стек протоколов** – перечень разноуровневых протоколов, реализованных в системе.

93) **Семейство протоколов TCP/IP** - Уровень доступа к сети (1-2) (специфицирует доступ к физической сети, на этом уровне **фреймы**), Межсетевой уровень (3) (в отличие от OSI, не устанавливает соединений с другими устройствами), Транспортный уровень (4-5) (обеспечивает доставку данных от компьютера к компьютеру, есть два протокола: TCP, UDP), Уровень прикладных программ (6-7) (в отличие от OSI, прикладные программы сами стандартизуют представление данных).

94) **IP-адрес** — это 32-разрядное число, которое кодирует информацию о номере конкретной сети и номере сетевого устройства этой сети.

95) **Протокол IP** – межсетевой протокол БЕЗ логического установления соединения. Протокол IP НЕ обеспечивает обнаружение и исправление ошибок. Он формирует дейтаграммы, поддерживает систему адресации, разбивает и обратно собирает дейтаграммы и организует маршрутизацию дейтаграмм.

96) **Дейтаграмма** – это пакет протокола IP.

97) **Шлюз** – устройство, передающее пакеты между различными сетями.

98) **Маршрутизация** – процесс выбора шлюза или маршрутизатора.

99) **Протокол контроля передачи (TCP, Transmission Control Protocol)** - обеспечивает надежную доставку данных с обнаружением и исправлением ошибок и с установлением логического соединения. (TCP обеспечивает последовательную передачу пакетов, контроль доставки пакетов, отработку сбоев)

100) **Протокол пользовательских дейтаграмм (UDP, User Datagram Protocol)** - отправляет пакеты с данными, не контролируя их доставку.

101) **Протоколы, опирающиеся на TCP**: TELNET (Network Terminal Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol)

102) Протоколы, опирающиеся на UDP: DNS (Domain Name Service), RIP (Routing Information Protocol), NFS (Network File System)

103) Требования к ОС: надежность (ошибки min), защита (от несанкционированного доступа), эффективность (удовлетворение критериям), предсказуемость (известны заранее проблемы и последствия различных действий). (НЕТ универсальных ОС)

104) Ядро (Kernel) — резидентная (постоянно находящаяся в ОП) часть ОС, работающая в режиме супервизора (т.е. может исполнять все множество команд ЦП). Обычно работает в режиме физической адресации.

105) Динамически подгружаемые драйверы устройств: резидентные/нерезидентные, работают в пользовательском/привилегированном режиме.

106) Системный вызов - обращение к ОС за предоставление той или иной функции (возможности, услуги, сервиса).

107) Подходы к структурной организации операционных систем: монолитное ядро, микроядерная архитектура (накладные расходы: один запрос распадается на множество подзапросов ввиду архитектуры).

108) Монолитное ядро - ядро, которое включает в себя все возможности операционной системы, запускается как единый процесс.

109) Микроядро - обеспечивает минимальные функции ОС: работа с адресным пространством, взаимодействие процессов, планирование.

110) Логические функции ОС: управление процессами, управление ОП, планирование (обработка очередей за обладание теми или иными ресурсами, обработка прерываний, распределение времени центрального процессора), управление устройствами и файловой системой, сетевое взаимодействие, безопасность (один пользователь в системе не мог добраться до инфы другого, плюс из-за открытых интерфейсов OSI,TCP/IP у компьютера появились “входы” через которые можно связывать компы и воровать данные).

111) Типы операционных систем: пакетная, разделения времени, реального времени.

112) Пакетная ОС – выполняет пакеты программ. Переключение с одного процесса на другой происходит только если: выполняемый процесс завершен, возникло прерывание, зацикливание процесса. (Критерий эффективности - максимальная загрузка процессора)

113) Квант времени ЦП – некоторый фиксированный ОС промежуток времени работы ЦП.

114) ОС Разделения времени - развитие модели пакетных ОС. Переключение с одного процесса на другой происходит по тем же причинам, как и в пакетной ОС + если исчерпался выделенный квант времени. (Критерий эффективности - минимизация времени отклика системы на запрос пользователя) В таких системах процессы делятся по критериям (например, интерактивные, счет-отладка, основной счет), для каждой из этих групп определяется квант времени и определяется приоритет для каждой из категорий процессов + эти приоритеты определяются по расписанию.

115) Системы реального времени - являются специализированными системами в которых все функции планирования ориентированы на обработку фиксированного набора событий за время, не превосходящее некоторого предельного времени. (процесс кипячение молока)

116) Сетевая ОС — ОС, которая обеспечивает функции распределения приложений в сети. (например почтовое приложение: клиентская часть, серверы и т.д.)

117) Распределённая ОС — ОС, функционирующая на многопроцессорном/многомашинном комплексе, в котором на каждом из узлов функционирует своё ядро, а также система, обеспечивающая распределение возможностей (ресурсов) ОС (т.н. сервисы или услуги). (пример функции - распределенная файловая система)

118) Процесс — совокупность машинных команд и данных, исполняющаяся в рамках ВС и обладающая правами на владение некоторым набором ресурсов.

119) Жизненный цикл процесса – образование процесса, обработка процесса, ожидание постановки на выполнение, завершение процесса.

120) Буфер ввода процессов (БВП) – пространство, в котором размещаются и хранятся сформированные процессы от момента их образования, до момента начала выполнения.

121) Буфер обрабатываемых процессов (БОП) — буфер для размещения процессов, находящихся в системе в мультипрограммной обработке.

122) «Полновесные процессы» — это процессы, выполняющиеся внутри защищенных участков оперативной памяти.

123) Легковесные процессы (нити) - не имеют собственных защищенных областей памяти. Они работают в мультипрограммном режиме одновременно с активировавшим их “полновесным” процессом и используют его виртуальное адресное пространство.

124) Организация процесса: однонитевая, многонитевая (в одном процессе).

125) Контекст процесса - совокупность данных, характеризующих актуальное состояние процесса. В Unix состоит из Пользовательской составляющей (тело процесса, состоит из сегмента кода и сегмента данных), Аппаратной составляющей (содержит все регистры и аппаратные таблицы ЦП, используемые активным или исполняемым процессом) и Системной составляющей (содержатся различные атрибуты процесса – pid, состояние, список областей памяти, таблицы открытых файлов, идентификаторы:, реальный идентификатор - идентификатор владельца файла, эффективный идентификатор - идентификатор пользователя, запустившего файл, сохраненные значения аппаратной составляющей, инфа о сигналах и т.п.)

126) Процесс в ОС Unix – объект, зарегистрированный в таблице процессов Unix или объект, порожденный системным вызовом fork().

127) Таблица процессов – содержит pid, часть контекста, ссылки на остальные части контекста.

128) Составляющие контекста, наследуемые при вызове fork(): окружение; файлы, открытые в процессе-отце; способы обработки сигналов; разрешение переустановки эффективного идентификатора пользователя; разделяемые ресурсы процесса-отца; текущий рабочий каталог и домашний каталог и т.п.

129) Составляющие контекста, ненаследуемые при вызове fork(): идентификатор процесса; идентификатор родительского процесса; сигналы, ждущие доставки в родительский процесс; время посылки ожидающего сигнала, установленное системным вызовом alarm(); блокировки файлов, установленные родительским процессом.

130) Замена тела при exec, **сохраняются**: pid, ppid, таблица дескрипторов файлов, приоритет и большинство атрибутов.

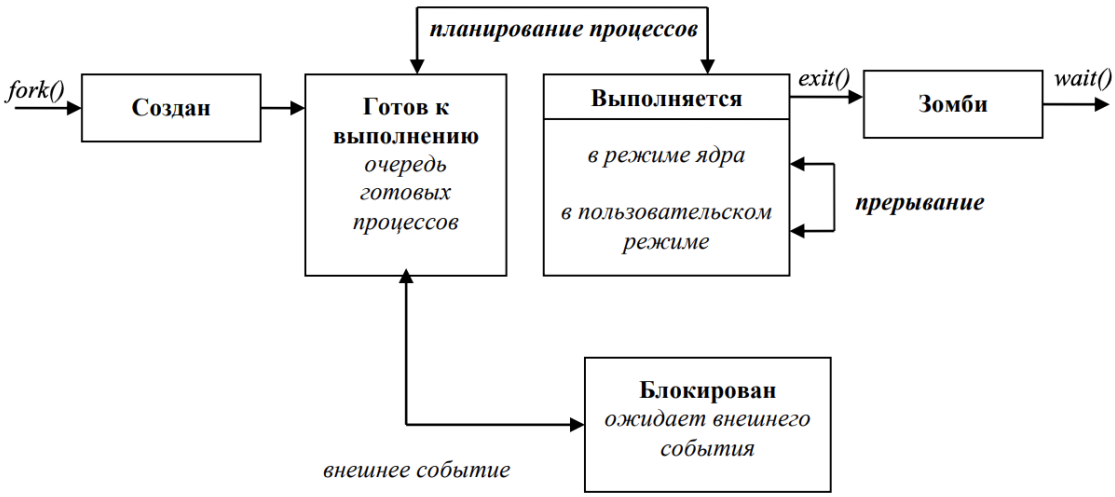
131) Замена тела при exec, **изменяются**: режимы обработки сигналов, эффективные идентификаторы владельца и группы, файловые дескрипторы (закрытие некоторых файлов)

132) Завершить процесс можно: системный вызов _exit(), выполнение оператора return в main() или выход на закрывающую скобку объемлющего блока функции, получение сигнала.

133) При завершении процесса: освобождаются сегмент кода и сегмент данных процесса, закрываются все открытые дескрипторы файлов, если у процесса имеются потомки, их предком назначается процесс с идентификатором 1, освобождается большая часть контекста процесса, однако сохраняется запись в таблице процессов и та часть контекста, в которой хранится статус завершения процесса и статистика его выполнения, процессу-предку завершаемого процесса посылается сигнал SIGCHLD.

134) Очистка зомби: используем wait(int * status)**,** старший байт - пользовательский код завершения, младший байт - системный код завершения, после этого все структуры, связанные с зомби освобождаются и удаляется запись из таблицы процессов.

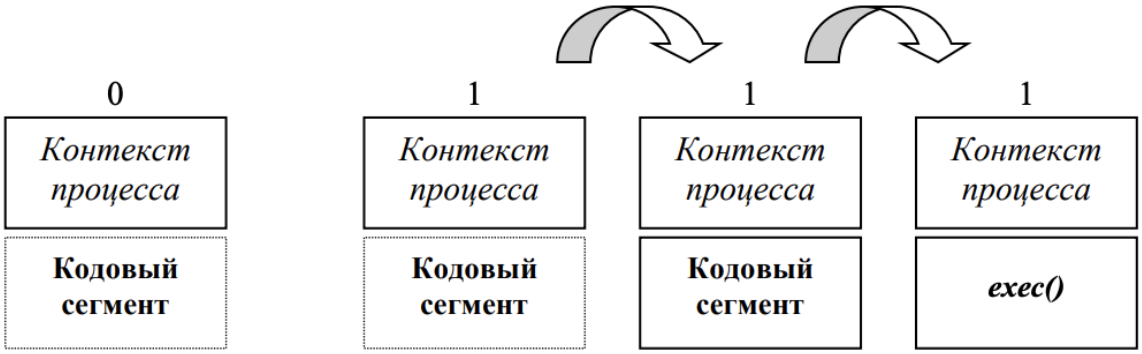
135) Жизненный цикл процессов ОС Unix:



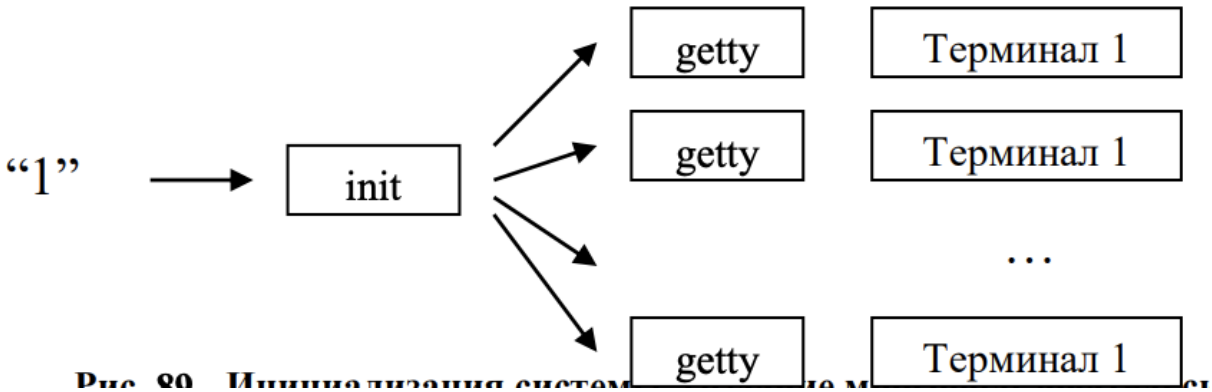
136) **Начальная загрузка** – это загрузка ядра системы в основную память и ее запуск. Состоит из нескольких этапов: 1. Аппаратный загрузчик читает нулевой блок системного устройства и передает точку входа. 2. После чтения этой программы она выполняется, т.е. ищется и считывается в память файл /unix, расположенный в корневом каталоге и который содержит код ядра системы. 3. Осуществляется запуск ядра операционной системы.

137) 0 процесс (некоторые действия по инициализации системы: инициализация аппаратных компонентов компьютера — начальная инициализация компонентов компьютера (настройка часов, инициализация контроллера памяти). Инициализация системных структур данных**)** не имеет кодового сегмента — это просто структура данных, используемая ядром

138) 1 процесс: (копируется 0 процесс, создается кодовый сегмент, копирование программы, реализующей системный вызов exec, запуск init, подключение интерпретатора команд к системной консоли, создание многопользовательской среды)



139) Getty — процесс, обеспечивающий работу конкретного терминала (то есть сеанс работы пользователя). Init создает процесс getty стандартным образом, и потом все все процессы создаются по схеме fork-exec.



140) Схема работы пользователя в ОС Unix:

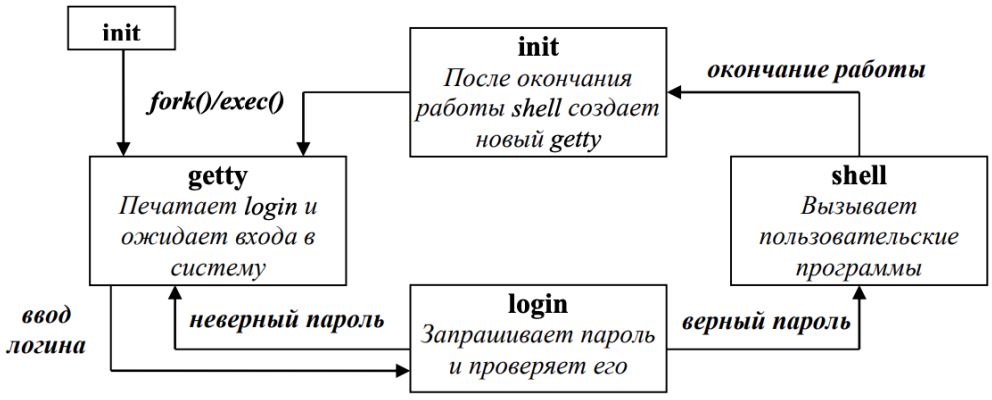


Рис. 90. Схема работы пользователя с ОС Unix.

141) Параллельные процессы - процессы, выполнение которых хотя бы частично перекрывается по времени.

142) Независимые процессы – процессы, использующие независимое множество ресурсов.

143) Взаимодействующие процессы совместно используют ресурсы, и выполнение одного может оказывать влияние на результат другого.

144) Разделение ресурса - совместное использование несколькими процессами ресурса ВС.

145) Критические ресурсы — разделяемые ресурсы, которые должны быть доступны в текущий момент времени только одному процессу.

146) Критическая секция, или критический интервал - часть программы (фактически набор операций), в которой осуществляется работа с критическим ресурсом.

147) Требование мультипрограммирования: результат выполнения процессов не должен зависеть от порядка переключения выполнения между процессами.

148) Гонка процессов (race condition) - ситуация, когда процессы конкурируют за разделяемый ресурс.

149) Взаимное исключение – способ работы с разделяемым ресурсом, при котором в тот момент, когда один из процессов работает с разделяемым ресурсом, все остальные процессы не могут иметь к нему доступ.

150) Тупик (deadlock) - ситуация, при которой из-за некорректной организации доступа и разделения ресурсов происходит взаимоблокировка.

151) Блокирование (дискриминация) — ситуация, когда доступ одного из процессов к разделяемому ресурсу не обеспечивается из-за активности других, более приоритетных процессов.

152) Способы реализации взаимного исключения (способы, которые позволяют работать с разделяемыми ресурсами: аппаратные и алгоритмические модели) - Запрещение прерываний и специальные инструкции, Алгоритм Петерсона, Активное ожидание, Семафоры Дейкстры, Мониторы Хоара, Обмен сообщениями.

153) Семафоры Дейкстры — формальная модель, предложенная голландцем Дейкстрой, которая основывается на следующем предположении: имеется тип данных, именуемой семафором (только 2 значения). Над семафором определенны 2 операции: увеличить на 1, уменьшить на 1. Процессы выбираются случайным образом, приоритетов нет.

154) Операций **атомарная**, если она не может быть прервана внешним воздействием ни в каком случае.

155) Двоичный семафор - семафор, максимальное значение которого равно 1.

156) Монитор Хоара — совокупность процедур и структур данных, объединенных в программный модуль специального вида. Структуры монитора доступны только для процедур, которые вошли в этот монитор. Процесс «входит» в монитор по вызову одной из его процедур. В любой момент времени внутри монитора может находиться не более одного процесса, чтобы понять, надо представить телефонную будку.

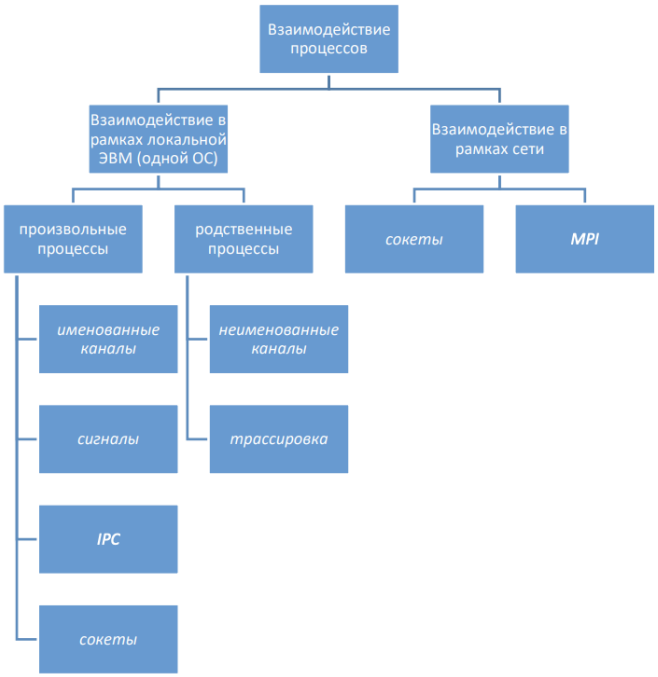
157) Аппарат обмена сообщениями. У него 2 функции: передачи данных и синхронизации. send(destination, source), receive(source, message). Для однопроцессорных систем и систем с общей памятью, для распределенных систем (когда каждый процессор имеет доступ только к своей памяти). Функции могут быть блокирующими и неблокирующими. Адресация может быть прямой и косвенной. Если блокирующий send, то отправитель заблокирован, пока получатель не получит, Блокирующий receive: получатель заблокирован пока не будет получено сообщение.

158) **Задача «Обедающие философы» (доступ равноправных процессов к разделяемому ресурсу)** - пять философов собираются за круглым столом, перед каждым из них стоит блюдо со спагетти, и между каждыми двумя соседями лежит вилка. Каждый из философов некоторое время размышляет, затем берет две вилки (одну в правую руку, другую в левую) и ест спагетти, затем опять размышляет и так далее. Каждый из них ведет себя независимо от других, однако вилок запасено ровно столько, сколько философов, хотя для еды каждому из них нужно две. Задача состоит в том, чтобы найти алгоритм, который позволит философам организовать доступ к вилкам таким образом, чтобы каждый имел возможность насытиться, и никто не умер с голоду.

159) **Задача «Читатели и писатели» (задача резервирования ресурса)** - Процессы-читатели считывают, а процессы-писатели записывают информацию в общую область памяти. Одновременно может быть несколько активных процессов-читателей. При записи информации область памяти рассматривается как критический ресурс для всех процессов, т. е. если работает процесс-писатель, то он должен быть единственным активным процессом. Задача состоит в определении структуры управления, которая не приведет к тупику и не допустит нарушения критерия взаимного исключения.

160) **Задача о «спящем парикмахере» (клиент-сервер с ограничением на длину очереди)** - Рассмотрим парикмахерскую, в которой работает один парикмахер, имеется одно кресло для стрижки и несколько кресел в приемной для посетителей, ожидающих своей очереди. Если в парикмахерской нет посетителей, парикмахер засыпает прямо на своем рабочем месте. Появившийся посетитель должен его разбудить, в результате чего парикмахер приступает к работе. Если в процессе стрижки появляются новые посетители, они должны либо подождать своей очереди, либо покинуть парикмахерскую, если в приемной нет свободного кресла для ожидания. Задача состоит в том, чтобы корректно запрограммировать поведение парикмахера и посетителей.

161) Взаимодействие процессов:



162) Сигнал – средство асинхронного уведомления процесса о наступлении некоторого события в системе.

163) kill(pid_t pid,int sig) , если pid=0, то сигнал будет передан всем процессам, которые принадлежат той же группе, что и процесс, посылающий сигнал, за исключением процессов с идентификаторами 0 и 1; если sig=0, то будет выполнена проверка корректности обращения к kill (в частности существование процесса с идентификатором pid), но никакой сигнал посылаться не будет; если процесс-отправителя не обладает правами привилегированного пользователя, то он может послать сигнал только процессам, у которых реальный или эффект pid владельца совпадает с реальным или эффект pid процесса-отправителя.

164) void (* signal (int sig, void (*disp)(int))) (int), int в disp - это номер отправляемого сигнала

165) Неименованный канал- область памяти(на диске), управляемой операционной системой, к которой не возможен доступ по имени, а только с помощью двух дескрипторов с ней ассоциированных. Область памяти, которая осуществляет выделение взаимодействующих процессам частей из этой области памяти для совместной работы, то есть это разделяемый ресурс. Fd[1] — запись в канал, fd[0] — чтение из канала. Для создания неименованного канала системный вызов pipe(). Если срабатывает, то возвращает 0. Порядок FIFO.

166) При чтении из неименованного канала: при чтении из пустого канала процесс блокируется до появления данных; если делается попытка прочитать больше данных, чем имеется в канале, то будет прочитано доступное количество данных; блокировка происходит лишь при условии, что есть хотя бы один открытый дескриптор записи в канал.

167) При записи в неименованный канал: если процесс пытается записать больше доступного объема в канал, то он записывает максимальное и блокируется до тех пор, пока не станет доступно достаточно памяти; если процесс пытается записать в канал больше предельного размера канала, то он блокируется и записывает при освобождении хотя бы 1 байта и так по циклу, пока не закончит; если пытается записать в канал, у которого не ассоциирован ни один дескриптор чтения, то он получает сигнал SIGPIPE.

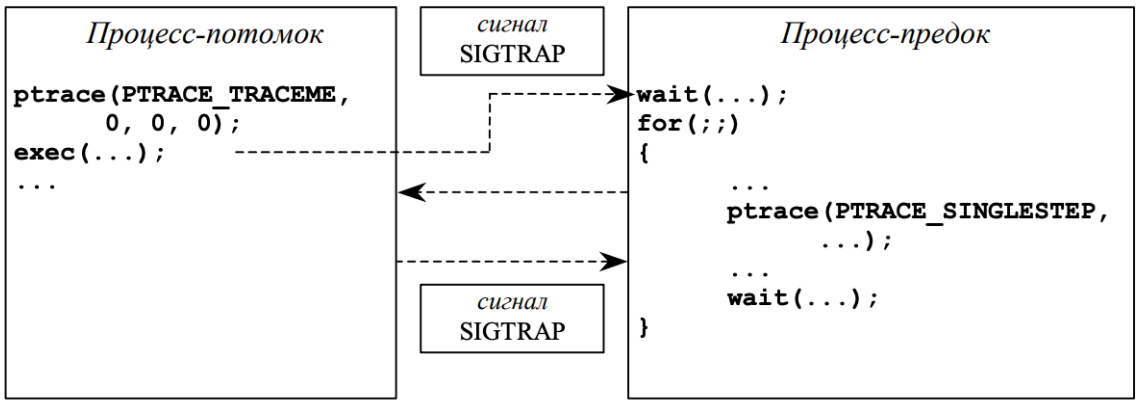
168) Конвейер – это две или более программ, которые исполняются параллельно и при этом стандартный вывод первой программы посылается на стандартный ввод второй программы, т.е. по мере того, как 1-й процесс генерирует свой вывод, он сразу же выдается на ввод второму процессу.

169) Именованные каналы – каналы, которые имеют имя, как и файлы. Каждому именованному каналу соответствует один элемент некоторого каталога ОС UNIX, поэтому возможна ссылка к нему по имени файла, которое хранится в поле имени соответствующего элемента каталога. Системный доступ реализован последовательно. У именованных каналов имеются имя владельца, права доступа, размер не ограничен. int mkfifo (char *pathname, mode_t mode), 0 -успех, -1 - ошибка. Если открыть на чтение (без записывающих процессов) или наоборот, то блокировка до появления, чтобы не было блокировки - O_NONBLOCK.

170) Трассировка — процесс пошагового выполнения программы. Выполнение отлаживаемого процесса-потомка приостанавливается всякий раз при получении им какого-либо сигнала, а также при выполнении вызова exec(). Если в это время отлаживающий процесс осуществляет системный вызов wait(), этот вызов немедленно возвращает управление. В то время, как трассируемый процесс находится в приостановленном состоянии, процесс-отладчик имеет возможность анализировать и изменять данные в адресном пространстве отлаживаемого процесса и в пользовательской составляющей его контекста. Далее, процесс-отладчик возобновляет выполнение трассируемого процесса до следующей приостановки (либо, при пошаговом выполнении, для выполнения одной инструкции).

171) ptrace(int cmd, int pid, int addr, int data) : cmd - код выполняемой программы, pid - идентификатор процесса-потомка, addr - некоторый адрес в адресном пространстве потомка, data - слово информации.

172) Общая схема трассировки процессов:



173) Отладчики бывают двух типов: **адресно-кодовыми** и **символьными**. Адресно-кодовые отладчики оперируют адресами тела отлаживаемого процесса, в то время как символьные отладчики позволяют оперировать объектами языка, т.е. переменными и операторами языка.

174) Установка breakpoint в адресно-кодовом: из адреса А кидает в таблицу отладчика, а туда ставим например деление на ноль, как приходим, проверяем, если это действительно тот адрес, то значит breakpoint, иначе ошибка. Потом вставляем машинное слово обратно и пошагово идем.

175) Установка breakpoint в символьном: там используется компилятор и редактор связей, так как нужна инфа, которую они собирают. Если статическая переменная - то берем адрес и отлаживаем. Если автоматическая - то в стек (контекст процесса). Если регистровая - то обращение к базе данных, сегменту кода.

176) Межпроцессное взаимодействие (Inter-Process Communication, IPC) — набор способов обмена данными между процессами. Состав: Очереди сообщений, Семафоры, Разделяемая память.

177) key_t fork (char* filename, char proj)

178) Очередь сообщений - хранилище типизированных сообщений, организованное по принципу FIFO. Любой процесс может помещать новые сообщения в очередь и извлекать из очереди имеющиеся там сообщения. Каждое сообщение имеет тип, представляющий собой некоторое целое число. (Команды: **msgget**(key,msgflg), msgflg: IPC_PRIVATE, IPC_CREAT, IPC_EXCL; **msgsend**(msgid,msgbuf,msgsz,msgflg), msgbuf состоит из long msgtype, char msgtext[] - тело сообщения, msgsz - размер тела сообщения, msgflg - 0 (блокировка, если недостаточно системных ресурсов) , IPC_NOWAIT; **msgrcv**(msgid,msgbuf,msgtype,msgflg), msgflg - IPC_NOWAIT (если сообщения в очереди нет), MSG_NOERROR (разрешение получать сообщение, даже если его длина больше ёмкости буфера, msgtype может быть 0 - любой, отрицательный - минимальный по модулю меньший); **msgctl**(msgid,cmd,struct msgid_ds *buf), cmd - IPC_STAT (скопировать в buf), IPC_SET (заменить на buf), IPC_RMID - удалить очередь может только процесс, у которого эффективный идентификатор совпадает с владельцем или создателем очереди), msg_id - структура, в полях которой хранятся права доступа к очереди, статистика обращений и т.д.)

179) Разделяемая память - механизм разделяемой памяти позволяет нескольким процессам получить отображение некоторых страниц из своей виртуальной памяти на общую область физической памяти. Данные, находящиеся в этой области памяти, будут доступны для чтения и модификации всем процессам, подключившимся к данной области памяти. (Команды: **shmget**(key,size,shmflag) , size - размер области памяти (если подключаемся, то меньше реальной должно быть); char * **shmat**(shmid, char* shmaddr, shmflg) - подключение области разделяемой памяти (возвращает адрес в виртуальном адресном пространстве, начиная с которого будет отображаться присоединяемая разделяемая память), shmaddr - адрес подключения, начиная с которого надо подключить разделяемую память (0 -автоматически или >=0), shmflag - например SHM_RDONLY, в случае успеха возвращает адрес, начиная с которого будет отображаться присоединяемая разделяемая память; **shmdt**(char *shmaddr), shmaddr - то что получили при shmat - отсоединение от разделяемой

памяти; **shmctl**(shmid,cmd,struct shmid_ds *buf), cmd - IPC_STAT, IPC_SET, IPC_RMID, SHM_LOCK,SHM_UNLOCK - блокировать или разблокировать область памяти, buf - управляющие параметры)

180) Семафоры представляют собой одну из форм IPC и используются для синхронизации доступа нескольких процессов к разделяемым ресурсам, т.е. фактически они разрешают или запрещают процессу использование разделяемого ресурса. (Команды: semget(key,nsems,semflg); semop(semid, struct sembuf * cmd_buf, size_t nops) , cmd_buf - массив из эл-тов типа sembuf, nops - кол-во элементов в массиве cmd_buf; semctl(semid,num,cmd,union semun arg), num - номера массива в массиве, cmd: IPC_SET - изменить параметры семафора, SETVAL - установить значение семафора, IPC_RMID, GETVAL - возвратить значение семафора, arg - управляющие параметры)

```
struct sembuf {
      short sem_num;  /* номер семафора в векторе */
      short sem_op;   /* производимая операция */
      short sem_flg;  /* флаги операции */
}
union semun {
     int val;  /* значение одного семафора */
     struct semid_ds *buf; /* параметры массива семафоров в
     целом */
     ushort    *array;   /* массив значений семафоров */
}
```

Если sem_op != 0, то
пока (val_sem_op < 0) [процесс блокирован]
val = val + sem_op
Если sem_op = 0, то
пока (val !=0) [процесс блокирован]
[возврат из вызова]

181) Сокеты — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Типы сокетов: **Соединение с использованием виртуального канала** - последовательный поток байтов, гарантирующий надежную доставку сообщений с сохранением порядка их следования; **Датаграммное соединение** - используется для передачи отдельных пакетов, содержащих порции данных - датаграмм (нет гарантий доставки в том же порядке), как правило датаграммное более быстрое

182) Функция создания сокета socket(int domain, int type, int protocol), domain — коммуникационный домен (AF_UNIX или AF_INET), type — тип сокета (SOCK_STREAM — виртуальный канал, SOCK_DGRAM — датаграммы), protocol — протокол для создания соединения в рамках данного домена (0 — автоматический выбор протокола, IPPROTO_TCP/UDP — протокол TCP/UDP).

183) Связывание сокета и имя, по которому к нему обращается клиент. int bind(int sockfd, struct sockaddr * myaddr, int protocol). Успешное связывание — 0, иначе -1. Клиент обязан знать имя сокета, а сокет, созданный на сервере,

не обязан знать имён клиентов. Т.е. с помощью команды bind() происходит связывание сокета с конкретным адресом.

184) Есть **сокеты с предварительным установлением соединения,** когда до начала передачи данных устанавливаются адреса сокетов отправителя и получателя, а есть **сокеты без установления соединение**, когда соединение до начала передачи данных не устанавливается, а адреса сокетов отправителя и получателя передаются с каждым сообщением. Если виртуалный канал - то 1 тип, если датаграмма - то как правило 2.

185) Прослушивание сокета. listen(int дескриптор сокета, максимальный размер очереди на соединение). Этот вызов используется процессором-сервером для того, чтобы сообщить системе о том, что он готов к обработке запросов на соединение, поступающих на данный сокет. Пока владелец сокета не вызовет listen(), все запросы на соединение с данным сокетом будут возвращать ошибку.

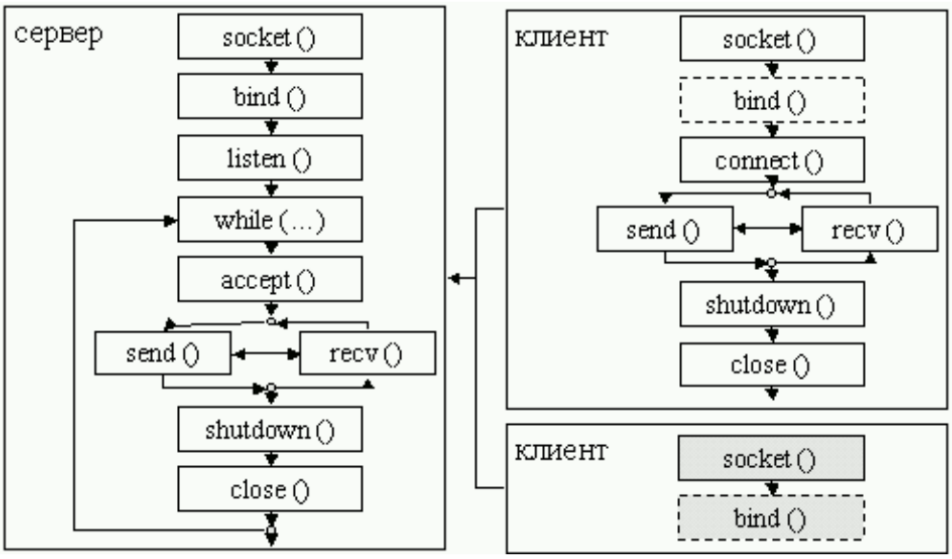
186) Запрос на соединение. connect(дескриптор сокета, указатель на структуру с адресом сокета, длина этой структуры). Это клиент говорит, что хочет установить соединение с сервером (его имя передаётся в параметре). Если не было listen, то connect работать не будет.

187) Подтверждение соединения. int accept(int дескриптор сокета, struct указатель на структуру с адресом сокета, int длина адреса). Этот вызов извлекает первый запрос из очереди запросов, ожидающих соединения, и устанавливает с ним соединение. Если к моменту вызова accept() очередь запросов на соединение пуста, процесс, вызвавший accept(), блокируется до поступления запросов. Пока не сработает accept, общения с клиентом не произойдёт. Accept() создает новый сокет, который будет использоваться для работы с данным соединением, и возвращает дескриптор этого нового сокета.

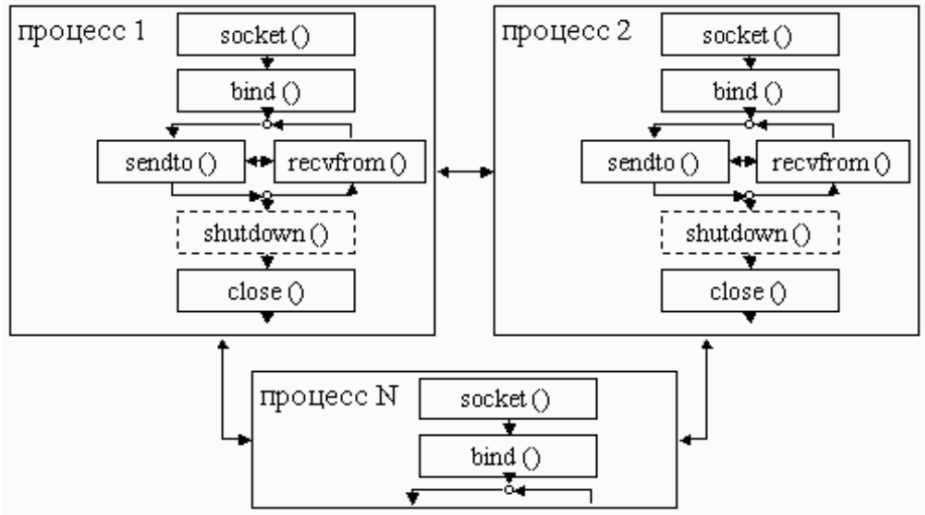
188) Получение и передача данных. int send (int sockfd, const void *msg, int len, unsigned int flags), int recv(int sockfdm void *buf,int len, unsugned int flags). Только для сокета с предварительно установленным соединением. Можно также использовать read и write. Флаги: MSG_OOB - прием/передача экстренных сообщений, MSG_PEEK - чтение порции данных без удаления их из сокета. Для сокетов без установления соединения можно использовать int sendto,int recvfrom.

189) Завершение работы с сокетом int shutdown(int sockfd, int mode): mode - 0 - закрытие на чтение, 1 - на запись, 2 - и то, и то. После этого еще надо сделать close- чтобы закрыть дескриптор файла.

190) Общая схема работы с сокетами с предварительным установлением соединения:



191) Общая схема работы с сокетами без предварительного установления соединения:



192) Файловая система — это компонента ОС, которая обеспечивает организацию хранения и доступ к данным пользователей посредством имен файлов и самих файлов.

193) Структурная организация файлов: последовательность байтов, последовательность записей произвольной длины, последовательность записей постоянной длины, иерархическая организация файла (дерево) - записи находятся в узлах дерева (запись состоит из двух полей: поле ключа и поле данных).

194) Атрибуты файла – имя, права доступа, персонификация (информация о создателе, владельце файла), тип файла (способ организации файла и интерпретация его содержимого: явная и неявная), размер записи (стационарный и нестационарный), размер файла, указатель чтения / записи, время создания, время последней модификации, время последнего обращения, предельный размер файла и т.д.

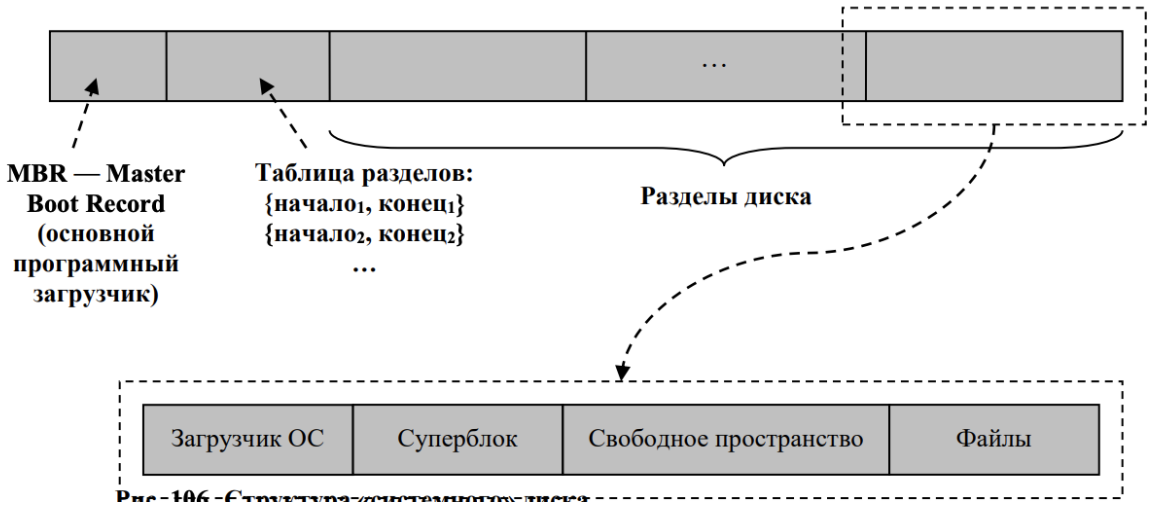
195) Файловый дескриптор – системная структура данных, содержащая информацию об актуальном состоянии «открытого» файла.

196) Основные сценарии работы с файлами: открытие, работа с содержимым или с атрибутами файлами, закрытие

197) Каталог – компонент файловой системы, содержащий информацию о содержащихся в файловой системе файлах. Каталог является специальным видом файлов.

198) Модели организации каталогов: одноуровневая файловая система (для бытовой техники), двухуровневая файловая система (для многопользовательской работы), иерархическая файловая система (**текущий каталог** - это каталог, на работу с которым в данный момент настроена ФС, **имя файла** - это имя файла относительно текущего каталога, **полное имя файла** - это перечень всех имен файлов от корня до узла с данным файлом, **относительное имя файла** - это путь от некоторого каталога до данного файла, **домашний каталог:** для каждого пользователя задается полное имя каталога, который должен стать текущим каталогом при входе пользователя в систему)

199) Практическая реализация файловой системы:



Суперблок – блок ФС, в котором находится информация о настройках ФС и актуальном состоянии ФС (информация о свободных блоках, данных, которые содержат каталоги.)

НЕ ПУТАТЬ: 1) блок физического HDD, 2) блок файловой системы и 3) блок файла! (расположены по уровням)

200) Модели реализации файлов: непрерывные файлы; файлы, имеющие организацию связанного списка; использование таблицы размещения файловой системы; индексные узлы(дескрипторы).

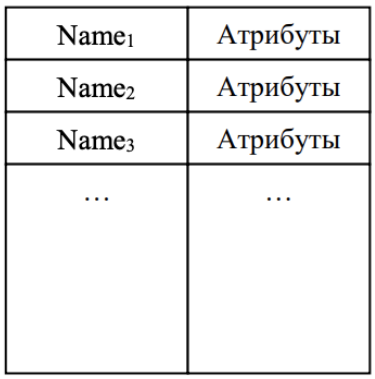
201) Непрерывные файлы: среди атрибутов надо: имя, блок начала и длина файл. Достоинства: простота реализации, высокая производительность. Недостатки: фрагментация свободного пространства, возможность увеличения размера существующего файла (при модификации).

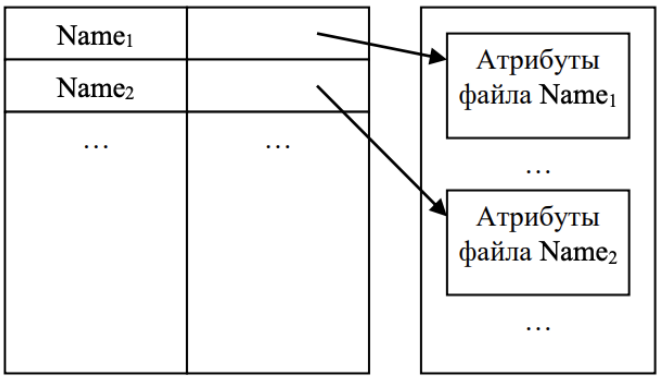
202) Файлы, имеющие организацию связанного списка: Достоинства: отсутствие фрагментации свободного пространства (за исключением блочной фрагментации), простота реализации, эффективный последовательный доступ. Недостатки: сложность организации прямого доступа, фрагментация файла по диску, наличие ссылки в блоке файла (ситуация чтения 2-х блоков при необходимости чтения 1-го блока)

203) Таблица размещения файловой системы(FAT): Кол-во строк в таблице совпадает с кол-вом блоков в файловой системе, и имеется каталог в котором для каждого имени файла имеется запись, содержащая номер начального блока. Достоинства: возможность использования всего блока для хранения данных файла, оптимизация прямого доступа (при полном или частичном размещении таблицы в ОЗУ). Недостатки: желательно размещение всей таблицы в ОЗУ.

204) Индексный узел (дескриптор) - системная структура данных, содержащая информацию о размещении блоков конкретного файла в файловой системе. Достоинства: нет необходимости в размещении в ОЗУ информации всей FAT обо всех файлах системы, в памяти размещаются атрибуты, связанные только с открытыми файлами. Недостатки: размер файла и размер индексного узла (в общем случае прийти к размерам таблицы размещения) - **решение:** ограничение размера файла, иерархическая организация индексных узлов (размещаются первые N блоков, а остальные в виде косвенной ссылки)

205) Организация каталогов: (во втором случае размеры атрибутов могут варьироваться)



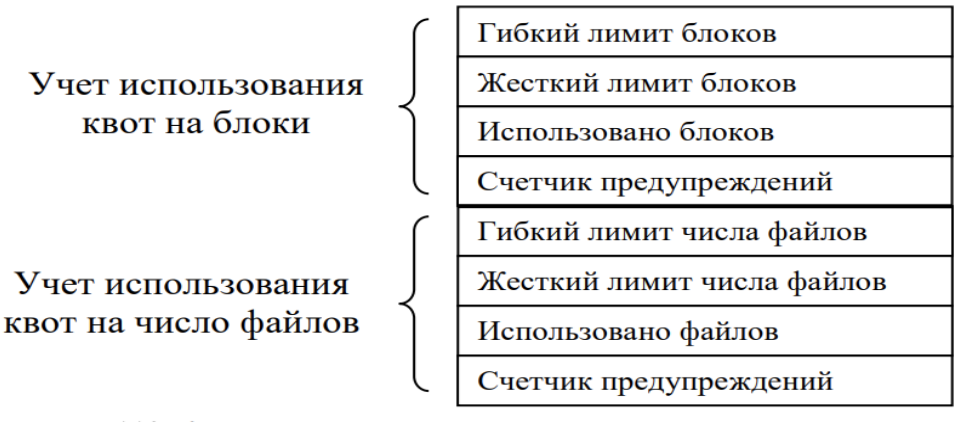


206) Соответствие имени файла и его содержимого: 1) Содержимому любого файла соответствует единственное имя файла; 2) Содержимому любого файла может соответствовать два и более имен файлов: жесткая связь (в атрибутах файлах новый счетчик имен, ссылающихся на этот файл) и символическая связь.

207) Координация использования пространства внешней памяти: Проблема - определение оптимального размера блока файловой системы. Если большой - эффективность обмена, но внутренняя фрагментация. Если маленький - эффективное использование пространства, фрагментация данных файла по диску.

208) Учёт свободных блоков ФС: связный список свободных блоков (в ОЗУ размещается первый блок списка), битовый массив (свободен - 1, занят - 0)

209) Квотирование пространства ФС:



Происходит учет кол-ва и размеров файлов конкретного пользователя. Жесткий лимит невозможно превысить. Если превысил гибкий лимит, то включается обратный счетчик предупреждений. Если он >0, то предупреждение, если 0 - то блокировка.

210) Надежность ФС: мб потери информации из-за сбоев или случайное удаление файлов => резервное копирование (архивирование):

1) избирательное копирование (копируются не все файлы ФС)
2) инкрементное архивирование (единожды создаем “полную” копию, а потом просто добавляем обновленные файлы)
3) использование компрессии (риск потери информации)
4) проблема “архивирования на ход” (при копировании начали работу с файлом)
5) распределенное хранение копий (в разных местах)

211) Стратегии архивирования:

1) физическая - “один в один”, модификация - интеллектуальное копирование (только занятых блоков => проблема обработки дефектных блоков)
2) логическая - копирование файлов (а не блоков), модифицированных после заданной даты

212) Проверка целостности ФС:

Контроль непротиворечивости блоков ФС: есть две таблицы (свободные и занятые блоки), сначала они обнуляются, а потом по свободным блокам и индексным узлам заносится туда информация (увеличивается на 1 в соответствующей таблице). Затем идет анализ и коррекция ситуаций: 1)непротиворечивость (они дополняют друг друга) , 2)пропавший (занятый) блок: либо оставить как есть, либо добавить в список свободных, 3)дубликат свободного блока - пересоздание списка свободных блоков, 4)дубликат занятого блока - автоматически решить нельзя, можем потерять инфу, действия следующие: а) копируются два файла, которые конфликтуют, б)старые удаляются, в)переопределение списка свободных блоков, г)обратное переименование файлов и фиксация факта их возможной проблемности.

Контроль непротиворечивости файлов ФС:

Пусть кол-во жестких связей к файлу = L, а в атрибуте файла счетчик M. Тогда: если L=M => OK, если L < M или L > M => счетчик = L

213) Файл Unix – это специальным образом именованный набор данных, размещенный в файловой системе.

214) Виды файлов: обычный, каталог, специальный файл устройств, именованный канал, ссылка, сокет.

215) Обычный файл (regular file) – традиционный тип файла, содержащий данные пользователя.

216) Каталог - файл данного типа содержит имена и ссылки на атрибуты, которые содержатся в данном каталоге.

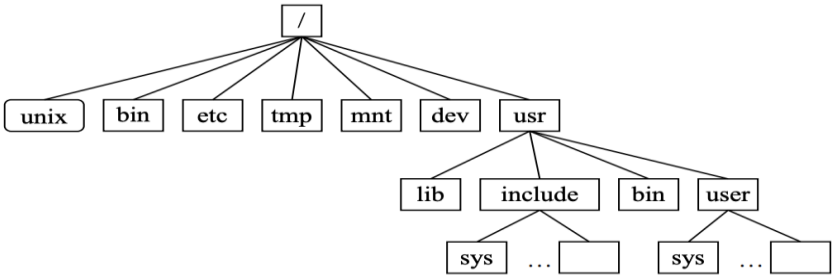
217) Специальный файл устройств (special device file) – система позволяет ассоциировать внешние устройства с драйверами и предоставляет доступ к внешним устройствам, согласно общим интерфейсам работы с файлами.

218) FIFO-файл - отдельный тип файла в файловой системе UNIX, который обладает всеми атрибутами файла, такими как имя владельца, права доступа и размер, но доступ к которому организован последовательно.

219) Ссылка (link) – позволяет создавать дополнительные ссылки к содержимому файла из различных точек файловой системы.

220) Права на доступ к файлу разделяются на три категории пользователей: владельца файла; права группы, к которой принадлежит владелец файла; права всех остальных пользователей системы.

221) Логическая структура каталогов:



/unix - файл загрузки ядра ОС

/bin - файлы, реализующие общедоступные команды системы

/etc - в этом каталоге содержатся системные таблицы и команды

/tmp - каталог для хранения временных системных файлов. При перезагрузке системы не гарантируется сохранение его содержимого

/mnt - каталог, к которому осуществляется монтирование дополнительных физических файловых систем для получения единого дерева логической файловой системы

/dev - каталог содержит специальные файлы устройств, с которыми ассоциированы драйверы устройств

/lib - здесь находятся библиотечные файлы языка Си и других языков программирования

/usr - размещается вся информация, связанная с обеспечением работы пользователей. Здесь также имеется подкаталог, содержащий часть библиотечных файлов (/usr/lib), подкаталог /usr/users (или /usr/home), который становится текущим при входе пользователя в систему, подкаталог, где находятся дополнительные команды (/usr/bin), подкаталог, содержащий файлы заголовков (/usr/include), в котором, в свою очередь, подкаталог, содержащий include-файлы, характеризующие работу системы (например, signal.h - интерпретация сигналов)

222) Модель ФС версии System V (s5fs):

Суперблок	Область индексных дескрипторов	Блоки файлов

Суперблок файловой системы содержит оперативную информацию о текущем состоянии файловой системы, а также данные о параметрах настройки (например, размер блока файла). Хранит информацию о свободных ресурсах ФС - свободных блоках и свободных индексных дескрипторах. Резидентно в ОП.

Индексный дескриптор - специальная структура данных ФС, которая ставится во взаимно однозначное соответствие с каждым файлом.

Блоки - свободные, занятые под системную информацию, занятые файлами.

223) Работа с массивами номеров свободных блоков:

если есть запрос на получение свободного блока, то ищем первую ячейку с содержательной (ненулевой) информацией => обнуляем ячейку и блок с найденным номеров выдаем в ответ на запрос

если блок освобождается, то действия в противоположном порядке

224) Работа с массивом свободных индексных дескрипторов:

Освобождение ИД: есть свободное место => номер -> элемент массива, нет свободного места - номер забывается.

Запрос ИД -> Поиск в массиве: массив пустой - и если в суперблоке есть информация о наличии свободных ИД => обновление массива, массив не пустой => изымается первый содержательный элемент

225) Индексный дескриптор хранит информацию о типе файла, правах доступа, информацию о владельце файла, размере файла в байтах, количестве имен, зарегистрированных в каталогах файловой системы и ссылающихся на данный индексный дескриптор. В частности, признаком свободного индексного дескриптора является нулевое значение последнего из указанных атрибутов. В индексном дескрипторе также собирается различная статистическая информация о времени создания, времени последней модификации, времени последнего доступа. И, наконец, в индексном дескрипторе находится массив блоков файла.

226) Адресация блоков файла: как пример массив блоков файла состоит из 13 элементов. Первые 10 используются для указания номеров первых десяти блоков файла, оставшиеся используются для организации косвенной адресации.

227) Файл-каталог: каждая запись в нем имеет фиксированный размер (пусть 16 байтов). Первые два байта хранят номер индексного дескриптора, оставишеся 14 - имя файла. При создании каталога он получает две записи, которые нельзя модифицировать или удалить - '.' и '..'

Так же есть установление связей между индексными дескрипторами и именами файлов: жесткая и символическая.

228) Достоинства System V:

1) Оптимизация в работе со списками номеров свободных ИД и блоков;
2) Организация косвенной адресации блоков файлов

229) Недостатки System V:

1) Концентрация важной информации в суперблоке;
2) Как следствие, проблема надежности (при потери суперблока);
3) Фрагментация файла по диску
4) Ограничения на возможную длину имени файла.

230) Основные задачи ОП: контроль состояния каждой единицы памяти, распределение, выделение, освобождение.

231) Стратегии и методы управления:

1) Одиночное непрерывное распределение
2) Распределение неперемещаемыми разделами
3) Распределение перемещаемыми разделами
4) Страничное распределение
5) Сегментное распределение
6) Сегментно-страничное распределение

232) Одиночное непрерывное распределение: это модель распределения оперативной памяти, прикол которой в том, что все адресное пространство подразделяется на 2 компонента. В одной части памяти располагается и функционирует ОС, а другая часть выделяется для выполнения прикладных процессов. "+" – примитивная простота, "-" – часть памяти, выделяемая под процесс, не используется, процессом память занимается все время выполнения, ограничение на размеры процесса.

233) Распределение неперемещаемыми разделами: это модель распределения оперативной памяти, прикол которой в том, что все адресное пространство подразделяется на 2 части. Одна часть отводится под ОС, а все оставшееся про-во под работу прикладных процессов, причем это пр-во заблаговременно делится на N частей (назовем их разделы), каждая из которых в общем случае имеет произвольный фиксированный размер. Эта настройка происходит на уровне ОС. Необходимые аппаратные средства: регистры границ, режим ОС. Очередь прикладных процессов разделяется по этим разделам. Существует 2 варианта организации этой очереди:

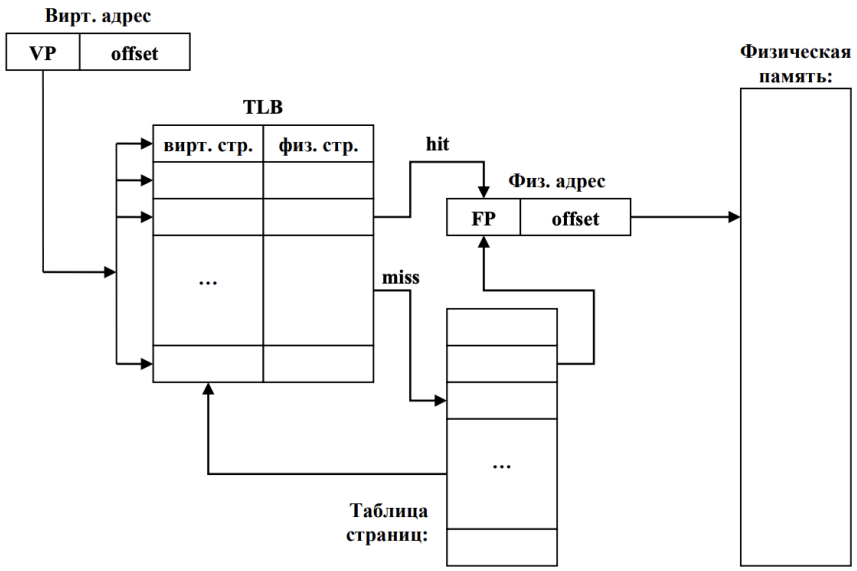
1) Присутствует только одна сквозная очередь, которая по каким-то соображениям распределяется между этими разделами. (Алг: процесс размещается в разделе минимального размера, достаточного для размещения)
2) С каждым разделом ассоциируется своя очередь и поступающий процесс сразу попадает в одну из этих очередей. (Алг 1: Освобождение раздела => поиск первого процесса, который может размещаться. Алг 2: Освобождение раздела => поиск процесса макс размера, который входит. Алг 3: модификация алг.2, но используем счетчик дискриминации, если больше предельного значения, то обходить нельзя)

“+” – простое средство организации мультипрограммирования, простые средства аппаратной поддержки, простые алгоритмы.
“-“ – фрагментация, ограничение размерами физической памяти, весь процесс размещается в памяти – возможно неэффективное использование.

234) **Распределение перемещаемыми разделами:** эта модель распределения разрешает загрузку произвольного числа процессов в оперативную память, и под каждый процесс отводится раздел необходимого размера, ну и также выделяется место под ОС. Система допускает перемещение раздела => и процесса. Такой подход позволяет избавиться от фрагментации.
После завершения тех или иных процессов пространство оперативной памяти все больше и больше освобождается места, т..е происходит фаргментация. Для борьбы с этим используют специальный процесс компрессии.
“+” – ликвидация фрагментации
“-“ – ограничение размером физической памяти, затраты на перекомпоновку.

235) **Страничное распределение:** эта модель распределения, которая предполагает разделение адресного пространства, как виртуального, так и физического на некоторые блоки фиксированного размера, которые называются страницами. Виртуальные страницы принадлежат каждому процессу. Физические страницы — это аппаратный ресурс.
“+” – отсутствие фрагментации памяти.
“-“ – размер таблицы страниц (любой процесс имеет собственную таблицу страниц), скорость отображения + аппаратные средства (полностью аппаратная таблица страниц, таблица страниц в ОЗУ + регистр начала таблицы страниц в памяти, гибридные решения)
Таблица страниц – это отображение номеров виртуальных страниц на номера физических.

236) **TLB (Translation Lookaside Buffer)** – таблица быстрого преобразования адресов.



Проблема - большой размер таблицы страниц. Дальше пойдут решения.

237) Решение - использование **иерархической таблицы страниц.**

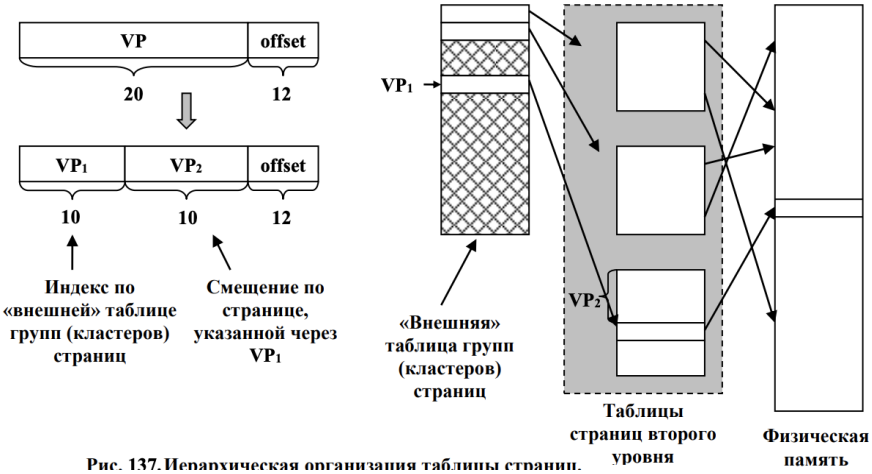
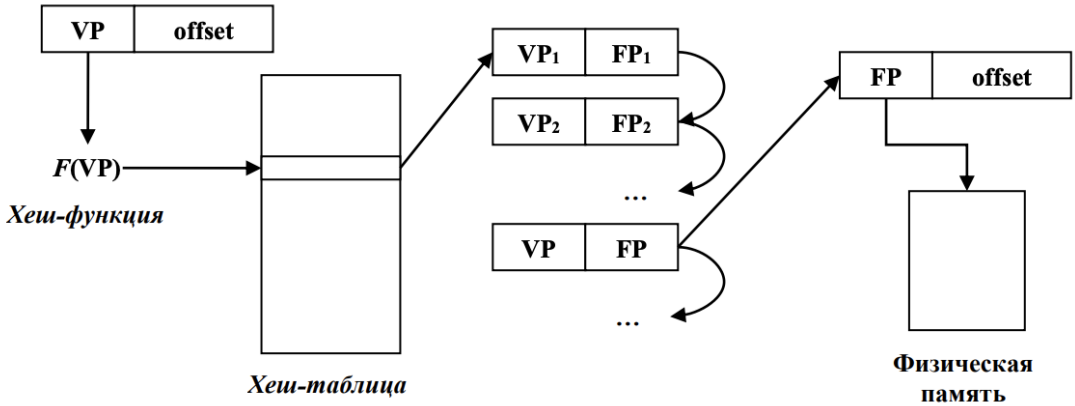
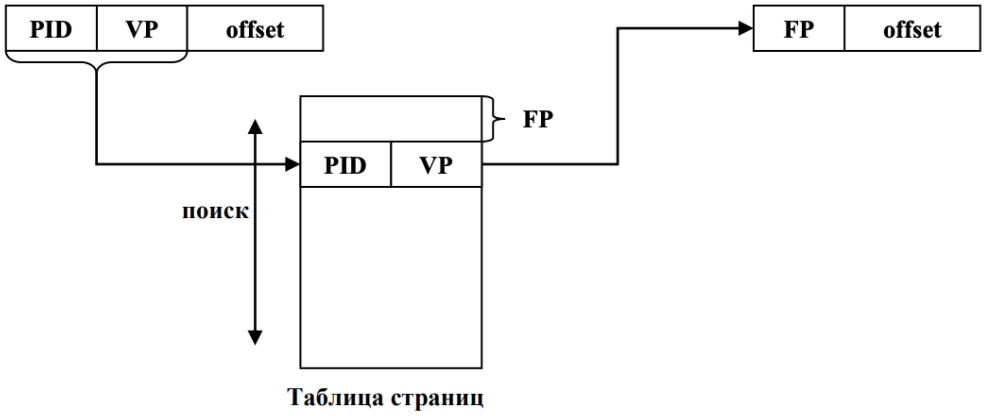


Рис. 137. Иерархическая организация таблицы страниц.

238) Хэш-таблицы:



239) Инвертированные таблицы страниц:



Плюс - единственная таблица страниц. Нет необходимости перегрузки таблицы при смене обрабатываемых процессов.

240) Плюс страничной организации: некоторые страницы процесса могут быть откачены во внешнюю память.

241) Проблема Замещения страниц: в связи с использованием страничной организации памяти возникает проблема выбора той страницы, которая должна быть откачена во внешнюю память (т.е. необходимо выбрать страницу для удаления из памяти), если возникла необходимость загрузить какую-то страницу из внешней памяти. Алгоритмы замещения:

242) Алгоритм NRU (Not Recently Used – не использовавшийся в последнее время) Используются биты статуса страницы. R – обращение, M – модификация. Устанавливаются аппаратно при обращении или модификации. Удаляется та страница, у которой минимальной кол-во единиц в этих битах в таком порядке: 00,01,10,11.

243) Алгоритм FIFO – first in, first out.

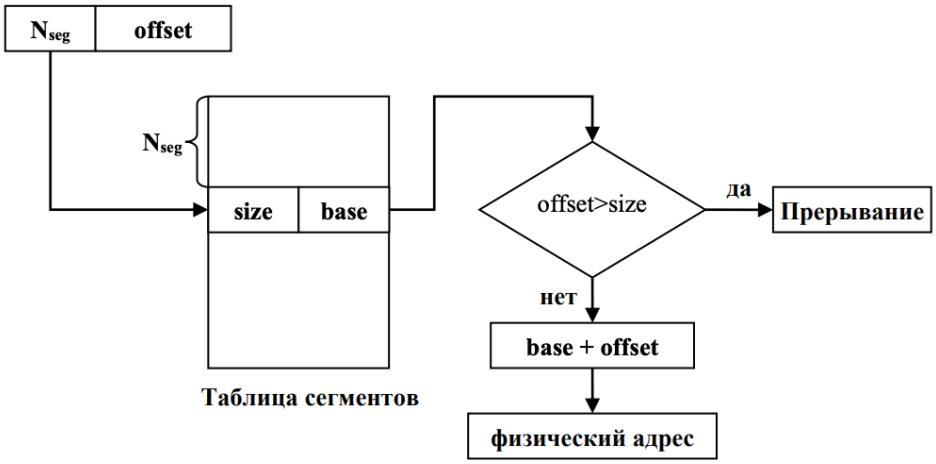
Модификация алгоритма (алгоритм вторая попытка): 1) выбирается самая старая страница. Если R=0, то она заменяется. 2)Если R=1, то R обнуляется, и страница переносится в конец очереди. Далее на п.1

244) Алгоритм Часы: 1) Если R=0, то выгрузка и стрелка на позицию вправо, 2)Если R=1, то R обнуляется, стрелка на позицию вправо, на п.1

245) Алгоритм NFU (Not Frequently Used – редко использовавшаяся страница): для каждой страницы счетчик. В начале счетчик = 0, по таймеру к счетчик прибавляется R. Удаляется страница с минимальным значением счетчика. Недостатки: помнит старую активность и возможно переполнение счетчика.

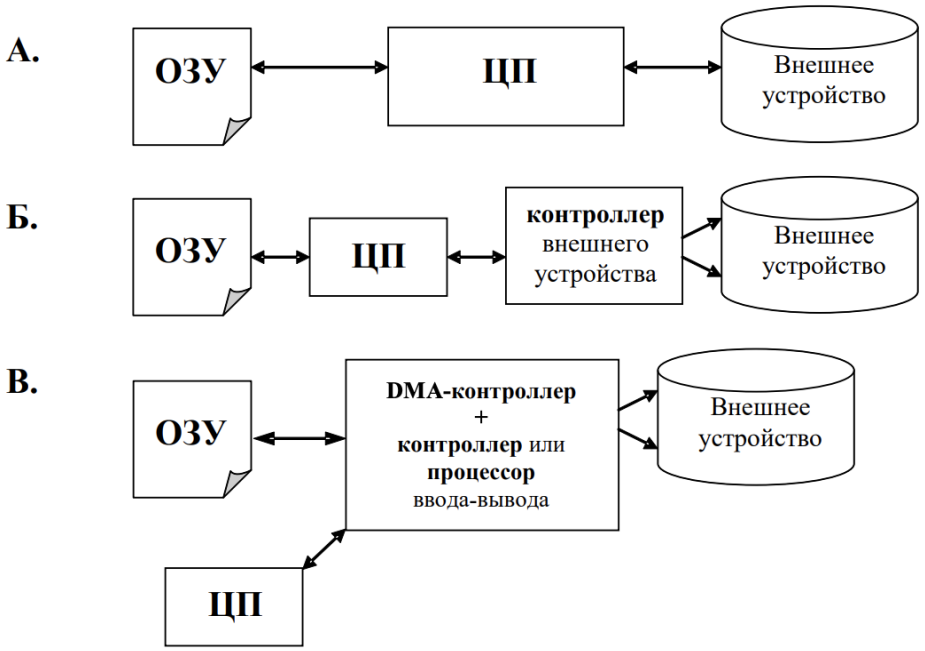
246) Модификация NFU (алгоритм старения): 1) значение счетчика сдвигается на один разряд вправо, 2) значение R добавляется в крайний левый разряд счетчика

247) Сегментная организация памяти: представляется в виде совокупности сегментов, где каждый из сегментов есть непрерывное пространство памяти и связанных с ним адресов. Каждый сегмент может иметь свою виртуальную адресацию. Он формируется, как: номер сегмента и смещение.



248) Сегментно-страничная организация памяти: концептуально это сегментная память, где каждый сегмент разделен на страницы.

249) Управление внешними устройствами. Архитектуры: непосредственное управление внешними устройствами ЦП, синхронное управление внешними устройствами с использованием контроллеров внешних устройств, асинхронное управление ВУ с использованием контроллеров внешних устройств, использование контроллера прямого доступа к памяти (DMA) при обмене, управление ВУ использованием процессора или канала ввода/вывода.



250) Программное управление внешними устройствами: унификация программных интерфейсов доступа к внешним устройствам, обеспечение конкретной модели синхронизации при выполнении обмена, выявление и локализация ошибок (а также устранение их последствий), буферизация обмена, обеспечение стратегии доступа к устройству, планирование выполнения операций обмена.

251) Алгоритмы планирования дисковых обменов:

FIFO - по очереди

LIFO - в обратном порядке

SSTF (Shortest Service Time First - “жадный алгоритм”) - на каждом шаге поиск обмена с минимальным перемещением

PRI - алгоритм, основанный на приоритетах процессов

SCAN (“лифтовый алгоритм”) - сначала движение в одну сторону до упора, затем в другую сторону до упора. Известно, что для любого набора запросов потребуется перемещений <= 2*число_дорожек_на_диске. Может привести к голоданию процессов.

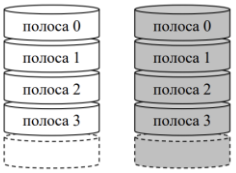
Путь головки	L
15 → 35	20
35 → 40	5
40 → 14	26
14 → 11	3
11 → 7	4
7 → 4	3
Итого: 61 Средний путь: 10,16	

C_SCAN - идем до минимального, а потом от максимального движемся вверх
N-step-SCAN - разделение очереди на подочереди длины <=N запросов каждая. последовательная обработка очередей, обрабатываемая очередь не обновляется, обновление очередей, отличных от обрабатываемой. Борьба с “залипанием” головки

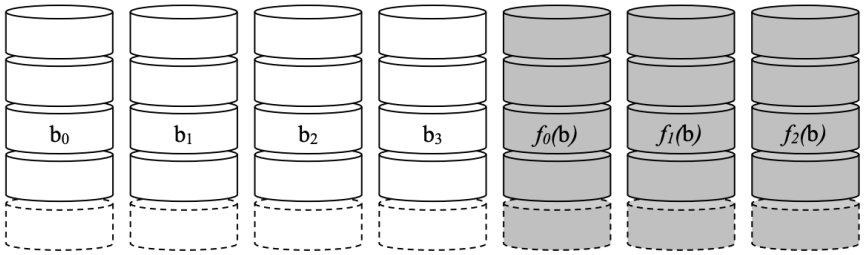
252) RAID система представляет собой набор независимых дисков, которые рассматриваются ОС как единое дисковое устройство, где данные представляются в виде последовательности записей, которые называются полосы.

RAID 0 - без избыточности

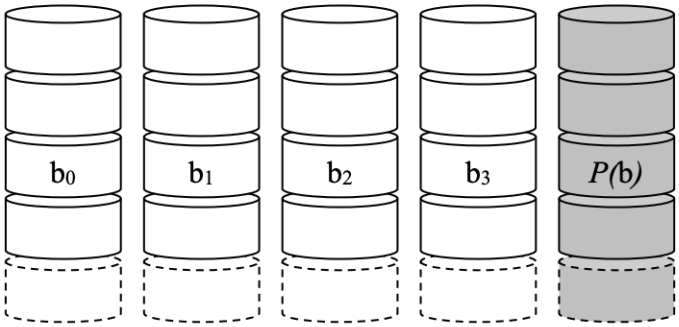
RAID 1 - зеркалирование



RAID 2 - избыточность с кодами Хэмминга => исправляет одну ошибку и выявляет две (синхронизированные головки, т.е. используются не независимые устройства)



RAID 3 - чётность с чередующимися битами. Для этого один из дисков назначается для хранения избыточной информации - полос, дополняющих до четности соответствующие полосы на других дисках (можно восстановить данные)



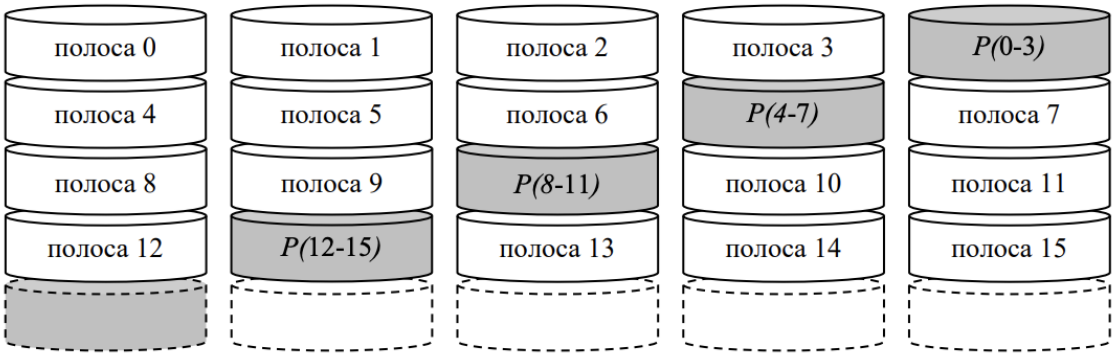
В данном случае имеется 4 диска данных и 1 диск четности. Тогда для диска четности:
$X_4(i)=X_0(i)xorX_1(i)xorX_2(i)xorX_3(i)$
Если произойдет потеря данных на первом диске, то для восстановления достаточно воспользоваться формулой:
$X_1(i)=X_0(i)xorX_2(i)xorX_3(i)xorX_4(i)$

RAID 4 - упрощение RAID 3 с несинхронизированными устройствами (проблема поддержания корректного состояния диска четности)

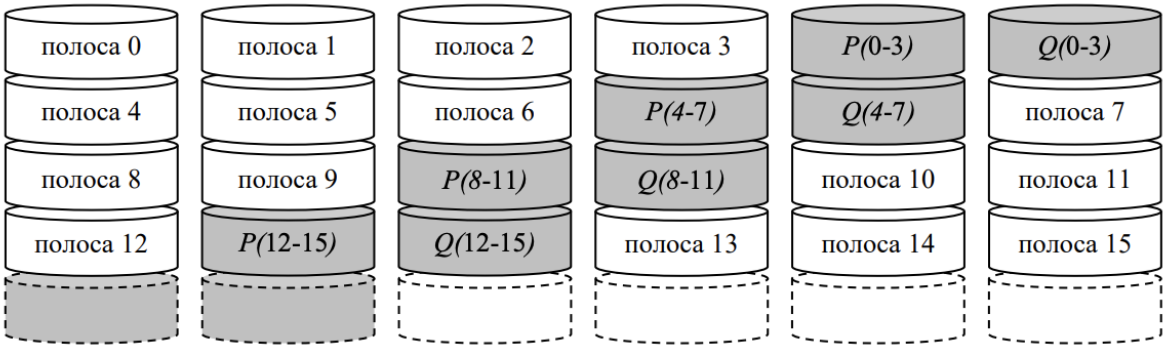
В данном случае имеется 4 диска данных и 1 диск четности. Тогда для диска четности:
$X_4(i)=X_0(i)xorX_1(i)xorX_2(i)xorX_3(i)$

После обновления полосы на первом диске:
$X_{4\ new}(i)=X_4(i)xorX_1(i)xorX_{1\ new}(i)$

RAID 5 - распределенная четность (циклическое распределение четности)



RAID 6 - двойная избыточность (циклическое распределение четности с использованием двух схем контроля; требуется N+2 дисков)



253) Файлы байториентированных устройств (драйверы обеспечивают возможность побайтного обмена данными и, обычно, не используют централизованной внутрисистемной кэш-буферизации)

254) Файлы блокориентированных устройств (обмен с данными устройствами осуществляется фиксированными блоками данных, обмен осуществляется с использованием специального внутрисистемного буферного кэша)

255) Содержимое файлов устройств размещается исключительно в соответствующем индексном дескрипторе.

256) Структура ИД файла устройства:

“старший номер” устройства - номер драйвера в таблице драйверов, соответствующей типу файла устройств

тип устройства (блок или байт)

“младший номер” устройства - некоторая дополнительная информация, передаваемая драйверу

257) bdevsw – таблица драйверов блокориентированных устройств

258) cdevsw - таблица драйверов байториентированных устройств

259) Запись таблицы - **Коммутатор устройства** – структура, в которой размещены указатели на соответствующие точки входа (функции) драйвера.

260) Ситуации, вызывающие обращение к функциям драйвера:

старт системы, определение ядром состава доступных устройств

обработка запроса ввода/вывода

обработка прерывания, связанного с данным устройством

выполнение специальных команд управления

261) Включение/удаление драйверов в систему:

1) “жесткое” встраивание драйверов код ядра

2) динамическое включение драйверов в систему: загрузка и динамическое связывание драйвера с кодом ядра, инициализация драйвера и соответствующего ему устройства

262) Организация обмена данными с файлами:

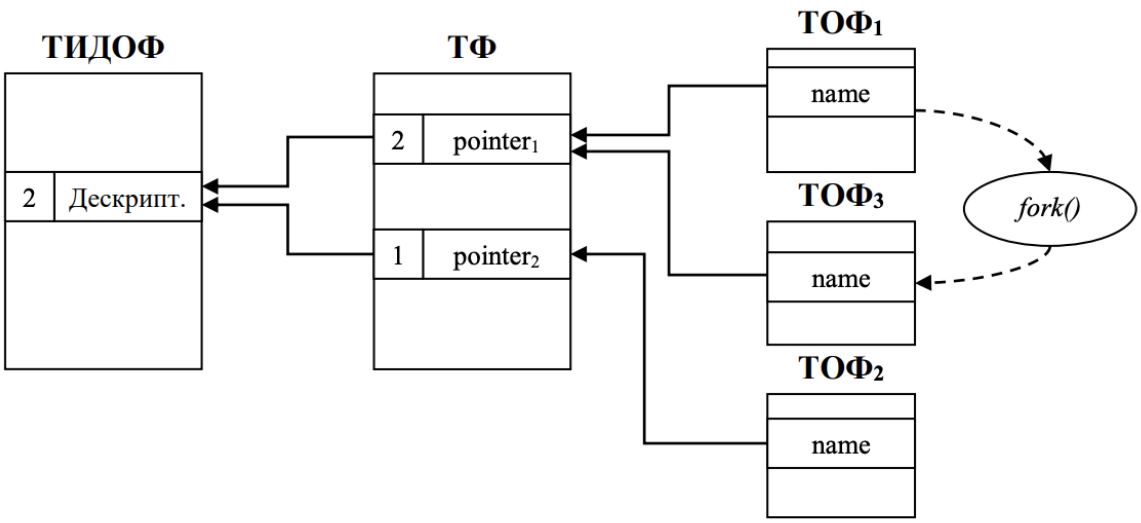


Таблица индексных дескрипторов открытых файлов (находится в ОП): для каждого открытого в рамках системы файла формируется запись в таблице ТИДОФ, содержащая: копии индексного дескриптора (ИД) открытого файла, кратность - счетчик открытых в системе файлов, связанных с данным ИД.

Таблица файлов (находится в ОП): таблица файлов содержит сведения обо всех файловых дескрипторах открытых в системе файлов

Таблица открытых файлов (в адресном пространстве процесса): С каждым процессом связана таблица открытых файлов (ТОФ). Номер записи в данной таблице есть номер ФД, который может использоваться в процессе. Каждая строка этой таблицы имеет ссылку на соответствующую строку ТФ.

263) Буферизация при блок-ориентированном обмене:

1. Поиск заданного блока в буферном пуле. Если удачно, то переход на п.4
2. Поиск буфера в буферном пуле для чтения и размещения заданного блока.
3. Чтение блока в найденный буфер.
4. Изменение счетчика времени во всех буферах.
5. Содержимое данного буфера передается в качестве результата.

“+” - оптимизация работы ОС за счет минимизации реальных обращений к физическому устройству

“-” - критичность к несанкционированному отключению питания, разорванность во времени факта обращения к системе за обменом и реальным обменом

264) Борьба со сбоями:

1) наличие параметра (его можно менять), определяющего периоды времени, через которые сбрасываются системные данные

2) пользовательская команда SYNC - сброс этой инфы по желанию пользователя

3) избыточность системы => возможность восстановить информацию