Операционные Системы

определения и основные понятия

1. **Вычислительная система** — совокупность аппаратных и программных средств, функционирующих в единой системе и предназначенных для решения задач определенного класса.
2. **Виртуальная машина** - совокупность всех средств, доступных программисту или пользователю для взаимодействия с ВС на различных уровнях доступа (аппаратном, физическом, логическом, систем программирования, прикладных систем).
3. **Драйвер физического устройства** — программа, основанная на использовании команд управления конкретного физического устройства и предназначенная для организации работы с данным устройством.
4. **Логическое** (**виртуальное**) **устройство** (**ресурс**) — устройство (ресурс), некоторые (возможно все) эксплуатационные характеристики которого реализованы программным образом.
5. **Драйвер логического** (**виртуального**) **ресурса** — программа, обеспечивающая существование и использование соответствующего ресурса.
6. **Ресурсы вычислительной системы** — совокупность всех физических и виртуальных ресурсов.
7. **Операционная система** — это комплекс программ, обеспечивающий контроль за существованием, распределением и использованием ресурсов ВС. Должна обладать свойствами: Надежность, Обеспечение защиты, Эффективность, Предсказуемость. Функции ОС: управление процессами, управление ОП, управление устройствами и ФС, планирование.
8. **Система программирования** — это комплекс программ, обеспечивающий поддержание жизненного цикла программы в вычислительной системе.
9. **Жизненный цикл программы** - этапы, связанные с разработкой и внедрением программы.
10. **Объектная среда** – это та ВС, в рамках которой продукт будет функционировать.
11. **Инструментальная среда** – это ВС, которая будет использована для разработки программ.
12. **Каскадная модель** - этапы проектирования, кодирования, тестирования и отладки выполняются последовательно и в одном направлении.
13. **Каскадная итерационная модель** - возможны возвраты на предыдущие этапы.
14. **Спиральная модель** – проект разбивается на последовательность промежуточных прототипов, для получения каждого проходим этап проектирования, кодирования, тестирования, отладки.
15. **Прототип** - программа, частично реализующая функциональность и внешний интерфейс разрабатываемой системы.
16. **Прикладная система** – программная система, ориентированная на решение или автоматизацию решения задач из конкретной предметной области.
17. **Принципы фон Неймана** - принцип двоичного кодирования, однородности памяти, адресуемости памяти, последовательного программного управления, условного перехода.
18. **Внешние устройства** – Внешние запоминающие устройства, Устройства ввода/отображения информации, Устройства приема и передачи данных.
19. **Устройство прямого доступа** - характеризуется возможностью чтения любой записи без предварительного просмотра каких-либо других записей.
20. **Устройство последовательного доступа** - для чтения i-й записи необходимо прочесть («просмотреть») все предыдущие.
21. **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.
22. **Рабочий цикл процессора** – последовательность действий, происходящая в процессоре во время выполнения программы.
23. **Устройство управления** – координирует выполнение команд программы процессором.
24. **Арифметико-логическое устройство** — выполнение команд, арифметическая или логическая обработка операндов.
25. **ОЗУ** – предназначено для хранения программы, выполняющейся в компьютере. ОЗУ состоит из ячеек памяти, содержащей поле машинного слова и поле служебной информации.
26. **Машинное слово** — поле программно изменяемой информации.

27. **Тег** (**Служебная информация**) — поле ячейки памяти, в котором автоматически размещается информация, необходимая для осуществления контроля над целостностью и корректностью использования данных, размещаемых в машинном слове. Может быть разряд четности, разряд команда-данные, разряд тип данных.
28. **Тип данных** – определенный формат данных, с конкретным набором операций, известных для этого формата.
29. **Производительность оперативной памяти** - скорость доступа процессора к данным, размещенным в ОЗУ. Характеризуется временем доступа и временем цикла.
30. **Время доступа** — время между запросом на чтение слова из оперативной памяти и получением содержимого этого слова.
31. **Время цикла памяти** — минимальное время между началом текущего и последующего обращения к памяти.
32. **Расслоение памяти** - вся память аппаратно разделена на последовательность областей, каждая из которых называется банк памяти и ОП физически представляется как объединение k банков памяти.
33. **Кэш-память** — Буферизация работы с ОП. Автоматически минимизирует число обращений к ОП. Обмен между КЭШем и ОП осуществляется блоками фиксированного размера.
34. **Буфер операндов** – аппаратная таблица, логически являющаяся компонентом ЦП (физически это может быть и отдельное от ЦП устройство), призванная аппаратно минимизировать количество обращений к «медленному» ОЗУ при записи и чтении операндов. Состоит из фиксированного числа строк вида [адрес/значение/признак изменения/код старения]
35. **Буфер команд** – минимизация обращений в ОЗУ за машинными командами. Состоит из фиксированного числа строк вида [адрес/значение /код старения]
36. **Сквозное кэширование** - при появлении команды записи менять содержимое соответствующего операнда в блоке КЭШа и сбрасывать этот блок в оперативную память.
37. **Кэширование с обратной связью** - при появлении команд записи меняется содержимое машинного слова только в КЭШе, при вытеснении модифицированного блока его содержимое сбрасывается в ОП.
38. **Прерывание** — событие в компьютере, при возникновении которого в процессоре происходит предопределенная последовательность действий. Короткое – не требует больших затрат. Фатальное – продолжение выполнения программы невозможно. Внутреннее – произошло в ЦП.
39. **Аппарат прерываний ЭВМ** - возможность аппаратуры ЭВМ стандартным образом обрабатывать возникающие в вычислительной системе события.
40. **Этап аппаратной обработки прерываний** – завершение текущей команды, блокировка прерываний, сохранение (частичное) состояния процессора, программная обработка прерывания.
41. **Иерархия памяти** – Регистры общего назначения и Кэш L1, Кэш L2, ОЗУ, ВЗУ прямого доступа с буферизацией, ВЗУ прямого доступа без буферизации, ВЗУ долговременного хранения.
42. **Регистровая память** — совокупность устройств памяти ЦП, предназначенных для временного хранения операндов, информации, результатов операций. Делится на Регистры общего назначения, Специальные регистры (Регистр адреса, Регистр результата, Слово состояния процессора, Регистр стека, Регистры внешних устройств) и Регистровые КЭШ-буферы.
43. **Регистровое окно** - один из способов решения проблемы вложенных процедур. Каждое окно состоит из области регистров, использующихся для получения и передачи параметров из/во внешние подпрограммы, области локальных регистров подпрограмм и области временных регистров. При вызове подпрограммы происходит переключение текущего регистрового окна на следующее регистровое окно. Области временных регистров могут пересекаться.
44. **Мультипрограммный режим** — режим, при котором возможна организация переключения выполнения с одной программы на другую. Необходимо: Аппарат защиты памяти, режим супервизора, аппарат прерываний.
45. **Аппарат защиты память** - аппаратная возможность ассоциирования некоторых областей ОЗУ с одним из выполняющихся процессов/программ.
46. **Аппарат виртуальной памяти** — аппаратные средства компьютера, обеспечивающие преобразование (установление соответствия) программных адресов, используемых в программе в адреса физической памяти, в которой размещена программа во время выполнения.
47. **Базирование адресов** — реализация одной из моделей аппарата виртуальной памяти. Выделяется регистр, в котором будет храниться адрес, начиная с которого размещается программа.

Исполняемые адреса, используемые в модуле будут автоматически преобразовываться в адреса физического размещения данных путем их сложения с этим регистром.

48. **Страничная память** – исполнительный физический адрес будет отличаться от исполнительного виртуального адреса за счет поля ”номер страницы”. В процессе выполнения при каждом обращении в память в адресе заменяется номер виртуальной страницы на номер соответствующей физической в соответствии с таблицей страниц. Страницы могут быть откачены на внешнюю память (свопинг).
49. **Сегментная память** - механизм организации виртуальной памяти, при котором виртуальное пространство делится на части произвольного размера — сегменты. Этот механизм позволяет, к примеру, разбить данные процесса на логические блоки.
50. **Сегменто-страничная организация памяти** - логический адрес состоит из трех полей: номера сегмента логической памяти, номера страницы внутри сегмента и смещения внутри страницы. Соответственно, используются две таблицы отображения – таблица сегментов, связывающая номер сегмента с таблицей страниц, и отдельная таблица страниц для каждого сегмента.
51. **Таблица страниц** — отображение номеров виртуальных страниц на номера физических. В системе имеется базовый регистр таблицы страниц, указывающий на таблицу страниц и хранящий ее длину. Каждый процесс имеет свой собственный набор таблиц страниц. Поля записей таблицы страниц: бит присутствие/отсутствие, поле защиты, признак изменения, обращение (чтение, запись, выполнение), признак блокировки кэширования.
52. **Иерархическая таблица страниц** – на примере двухуровневой. Виртуальный адрес состоит из трех полей [p1,p2,s]. В памяти хранится таблица, в которой по номеру p1 находится адрес другой таблицы, в которой по номеру p2 находится адрес страницы. Можно хранить в памяти лишь активную часть таблиц второго уровня, тем самым экономя память. Двухуровневая таблица требует 2x дополнительных обращения к памяти.
53. **Инвертированная таблица страниц** - предназначена для сокращения размеров таблиц страниц. В таблице страниц хранится один элемент для каждой реальной страницы, находящейся в памяти, за счет этого экономится память. Виртуальный адрес по-прежнему состоит из номера страницы и смещения в ней: [p,s]. При каждом обращении к памяти процессор к номеру страницы прибавляет номер процесса [pid,p]. Именно такие пары хранятся в инвертированной таблице страниц. Пробема – долгий поиск в такой таблице. Преимущество – не нужно ее менять при смене процесса, т.к. страницы уже ассоциированы с процессами, процесс не попадет в чужую область.
54. TLB (Translation Lookaside Buffer) – для того, чтобы не делать обращения в медленную память при каждой адресации, есть буфер быстрого преобразования адресов, который используется в качестве КЭШ таблицы страниц.
55. **Алгоритмы замещения страниц** - NRU (Not Recently Used), FIFO, Second-chance, LRU (Least Recently Used), NFU (Not Frequently Used).
56. **Виртуальное адресное пространство** – множество виртуальных страниц, доступных для использования в программе. Количество виртуальных страниц определяется размером поля «номер виртуальной страницы» в адресе.
57. **Физическое адресное пространство** – оперативная память, подключенная к данному компьютеру. Физическая память может иметь произвольный размер (число физических страниц может быть меньше, больше или равно числу виртуальных страниц).
58. **Классификация Флинна** - Считаем потоки данных и команд независимыми (условно). SISD, SIMD, MISD, MIMD.
59. SISD — Single Instruction, Single Data stream — компьютер с единственным ЦП, обрабатывающий одну порцию данных.
60. SIMD — Single Instruction, Multiple Data — матричная обработка данных.
61. MIMD — множество процессоров одновременно выполняют различные последовательности команд над своими данными. С общей оперативной памятью: UMA, NUMA, SMP. С распределенной оперативной памятью: MMP, COW.
62. UMA - Uniform Memory Access — характеристики доступа любого процессорного элемента в любую точку ОЗУ не зависят от конкретного элемента и адреса (Все процессоры равноценны относительно доступа к памяти).
63. NUMA - Non-Uniform Memory Access - процессорные элементы работают на общем адресном пространстве, но характеристики доступа процессора к ОЗУ зависят от того, куда он обращается.

64. COW - Claster of Workstations - многомашинная система, машины в которой объединены специальной быстрой сетью. Кластер из машин разной архитектуры и производительности – гетерогенный.
65. MMP – Massively Parallel Processors - Промышленное развитие кластеров. Используются спец. средства коммуникации, более дорогие и более специализированные.
66. **Терминальный комплекс** — многомашинная ассоциация, предназначенная для организации массового доступа удаленных и локальных пользователей к ресурсам некоторой вычислительной системы.
67. **Хост** - абонентский или основной компьютер.
68. **Коммутируемые каналы** - каждый раз выбирается новый маршрут.
69. **Выделенные каналы** - обеспечивает связь на постоянной основе.
70. **Симплексные каналы** - передача информации ведется в одном направлении.
71. **Дуплексные каналы** - одновременная передача информации в двух направлениях.
72. **Компьютерная сеть** — объединение компьютеров (вычислительных систем), взаимодействующих через коммуникационную среду.
73. **Коммуникационная среда** — каналы и средства передачи данных.
74. **Сообщение** — логически целостная порция данных, имеющая произвольный размер.
75. **Сеть коммутации каналов** - обеспечивает установку канала связи на время всего сеанса связи между абонентскими машинами.
76. **Сеть коммутации сообщений** - сеанс связи представляется в виде последовательности сообщений. Сообщение – это порция данных произвольного размера. Сообщение отправляется в сеть по некоторой информации о маршруте.
77. **Сеть коммутации пакетов** - Все сообщения разделяются на блоки данных некоторого фиксированного размера. После этого сеть работает так же, как сеть коммутации сообщений, но с пакетами. Каждая машина пытается от пакета избавиться (принцип горячей картошки).
78. **Модель** ISO OSI – система открытых интерфейсов, состоит из 7 уровней: Физический, Канальный, Сетевой, Транспортный, Сеансовый, Представительский, Прикладной. 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).
79. **Семейство протоколов** TCP/IP - Уровень доступа к сети (1-2), Межсетевой уровень (3), Транспортный уровень (4-5), Уровень прикладных программ (6-7).
80. **Протокол** UDP - User Datagram Protocol – транспортный протокол для передачи данных в сетях IP без установления соединения. В отличие от TCP, UDP не гарантирует доставку пакета.
81. **Протокол** TCP - Transmission Control Protocol - это транспортный протокол, предоставляющий поток данных, с предварительной установкой соединения, за счёт этого дающий уверенность в достоверности получаемых данных, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета. В отличие от UDP гарантирует, что приложение получит данные точно в такой же последовательности, в какой они были отправлены, и без потерь.
82. **Протокол** IP – межсетевой протокол без логического установления соединения. Он не обменивается контрольной информацией для установки соединения. IP оставляет другим протоколам право устанавливать соединения – этим занимается либо протокол TCP, либо сами прикладные программы.

    Протокол IP не обеспечивает обнаружение и исправление ошибок
83. **Сетевое устройство** – устройство, с которым ассоциирован некий стек протоколов.
84. **Система адресации протокола** IP - IP-адрес состоит из двух частей: номера сети и номера узла. Если весь IP-адрес состоит только из двоичных нулей, то он обозначает адрес того узла, который сгенерировал этот пакет. Если все двоичные разряды IP-адреса равны 1, то пакет с таким адресом назначения должен рассылаться всем узлам, находящимся в той же сети. Если в поле номера узла назначения стоят только единицы, то пакет, имеющий такой адрес, рассылается всем узлам сети с заданным номером сети.
85. **Протокол** — формальное описание сообщений и правил, по которым сетевые устройства (вычислительные системы) осуществляют обмен информацией. Правила взаимодействия одноимённых (одноранговых) уровней сети.
86. **Интерфейс** — правила взаимодействия вышестоящего уровня с нижестоящим.

87. **Служба** (**сервис**) — набор операций, предоставляемых нижестоящим уровнем вышестоящему.
88. **Стек протоколов** — перечень разноуровневых протоколов, реализованных в системе.
89. **Пакет** – это блок данных, который передаётся вместе с информацией, необходимой для его корректной доставки. Каждый пакет перемещается по сети независимо от остальных.
90. **Фрейм** (**Кадр**) — фрагмент данных сетевого протокола канального уровня модели OSI, передаваемый по линии связи.
91. **Дэйтаграмма** — пакет протокола IP. Блок информации, посланный как пакет сетевого уровня через передающую среду без предварительного установления соединения и создания виртуального канала.
92. **Шлюз** — устройство, передающее пакеты между различными сетями. Дейтаграммы, которые переправляет шлюз, поднимаются в нем только до межсетевого уровня. На этом уровне шлюз, узнавая адрес получателя данных, принимает решение отправить дейтаграмму в одну из сетей, к которым подключен.
93. **Маршрутизация** — процесс выбора шлюза или маршрутизатора.
94. **Локальные мультиплексоры** – аппаратные комплексы, предназначенные для осуществления связи и взаимодействия вычислительной системы с несколькими устройствами через один канал ввода/вывода.
95. **Локальные терминалы** – оконечные устройства, используемые для взаимодействия пользователей с вычислительной системой, подключаемые к вычислительной системе непосредственно через каналы ввода/вывода или через локальные мультиплексоры.
96. **Модемы** – устройства, предназначенные для организации взаимодействия вычислительной системы с удаленными терминалами с использованием телефонной сети.
97. **Удаленные терминалы** – терминалы, имеющие доступ к вычислительной системе с использованием телефонных линий связи и модемов.
98. **Удаленные мультиплексоры** – мультиплексоры, подключенные к вычислительной системе с использованием телефонных линий связи и модемов.
99. **Программа** – просто текст (может быть, скомпилированный)
100. **Процесс** — совокупность машинных команд и данных, исполняющаяся в рамках ВС и обладающая правами на владение некоторым набором ресурсов. (Существует много определений процесса. Например: элементарная программная единица, которой осуществляет управление ОС.)
101. **Буфер ввода процессов** (**БВП**) – пространство, в котором размещаются и хранятся сформированные процессы от момента их образования, до момента начала выполнения.
102. **Буфер обрабатываемых процессов** (**БОП**) — буфер для размещения процессов, находящихся в системе в мультипрограммной обработке.
103. **Жизненный цикл процесса** – Ожидает начала обработки ЦП, Обрабатывается ЦП, Ожидает завершения операций ввода/вывода, Ожидает продолжения обработки ЦП, Обрабатывается ЦП, Завершен.
104. **Разделяемые ресурсы** — ресурсы, которые могут одновременно принадлежать двум или более процессам.
105. **Ядро** (Kernel) — резидентная (постоянно находящаяся в памяти) часть ОС, работающая в режиме супервизора (т.е. может исполнять все множество команд ЦП). Обычно работает в режиме физической адресации.
106. **Монолитное ядро** - ядро, которое включает в себя все возможности операционной системы, запускается как единый процесс.
107. **Микроядро** - обеспечивает минимальные функции ОС: работа с адресным пространством, взаимодействие процессов, планирование.
108. **Системный вызов** — средство ОС, обеспечивающее возможность процессов обращаться к ОС за теми или иными функциями.
109. **Квант времени ЦП** — некоторый фиксированный ОС промежуток времени работы ЦП. Непрерывный период процессорного времени.
110. **Пакетная ОС** – выполняет пакеты программ. Переключение с одного процесса на другой происходит только если: Выполняемый процесс завершен, Возникло прерывание по обмену в выполняемой программе, Зафиксировался факт зацикливания.
111. **Система разделения времени** - переключение выполнения процессов происходит только в одном из случаев: Исчерпался выделенный квант времени, Выполнение процесса завершено, Возникло прерывание, Был фиксирован факт зацикливания процесса.

112. **Система реального времени** - все функции планирования ориентированы на обработку некоторых событий за время, не превосходящее некоторого предельного значения.
113. **Сетевая ОС** — ОС, которая обеспечивает функции распределения приложений в сети
114. **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.
115. **Управление внешними устройствами** - Непосредственное управление (поток команд и поток данных идут через ЦП), Синхронное управление (появляется контроллер ВУ, берет на себя часть функций ЦП), Асинхронное управление (с появлением аппарата прерываний).
116. **Контроллеры** DMA - Direct memory access – позволяют вывести поток данных за пределы ЦП. Поток управляющей информации остается в ведении ЦП.
117. **Процессоры ввода-вывода** - позволяют обеспечить высокоуровневый интерфейс для ЦП при управлении внешними устройствами. ЦП предоставляются различные макрокоманды (например «записать на диск … начиная с …места»).
118. **Программное управление внешними устройствами** – Драйверы логических устройств, Драйверы физический устройств, Программы обработки прерываний, Аппаратура.
119. **Схемы буферизации ввода-вывода** – Без буферизации, Одинарная буферизация, Двойная буферизация, Циклическая буферизация.
120. **Планирование дисковых обменов** - Случайная выборка, Модель FIFO, Модель LIFO, SSTF, Приоритетные алгоритмы (последовательность обменов имеет характеристику приоритетов), SCAN, C-SCAN, N-step-SCAN.
121. SSTF (shortest-seek-time-first) - первым обслуживается запрос, который характеризуется минимальным расстоянием подвода (и тем самым наименьшим временем поиска цилиндра), если даже этот запрос не является первым в очереди. Для стратегии SSTF характерна резкая дискриминация определенных запросов.
122. SCAN - Стратегия SCAN в общем аналогична SSTF, если не считать того, что она выбирает для обслуживания тот запрос, для которого характерно минимальное расстояние поиска в привилегированном направлении. Если в текущий момент привилегированное направление — это от внутренних дорожек к наружным, то стратегия SCAN выбирает запрос с минимальным расстоянием подвода в наружном направлении. При реализации стратегии SCAN каретка с головками не меняет направления своего движения до тех пор, пока она не достигнет самого наружного цилиндра или пока не выяснится, что больше нет запросов, ожидающих обслуживания при движении в текущем привилегированном направлении.
123. N-step-SCAN - N-шаговое сканирование, головки также совершают движения туда и обратно, как в случае SCAN, но на каждом проходе обслуживаются только те запросы, которые существовали в момент начала прохода. Запросы, поступающие во время прохода, группируются и упорядочиваются таким образом, чтобы их можно было оптимально обслужить на обратном ходу.
124. C-SCAN - «циклическое сканирование», исключает дискриминационное отношение к внутренним и наружным цилиндрам. Магнитные головки перемещаются от наружного цилиндра к внутреннему, причем обслуживание запросов производится по наикратчайшему времени поиска. Когда каретка завершает свой прямой ход, она скачком возвращается на обслуживание запроса, ближайшего к самому внешнему цилиндру, а затем возобновляет обслуживание запросов на прямом ходе, и внутреннему цилиндру. Стратегию C-SCAN можно реализовать таким образом, чтобы запросы, поступающие во время текущего прямого хода, обслуживались при следующем ходе, благодаря этому стратегия C-SCAN полностью исключает дискриминацию запросов к внутренним или наружным цилиндрам.
125. RAID **системы** - redundant array of independent disks — избыточный массив независимых жёстких дисков - набор физических дисковых устройств, рассматриваемых операционной системой, как единое дисковое устройство. RAID 0 (ДМ повышенной производительности и меньшей отказоустойчивости), RAID 1 (зеркальный ДМ), RAID 2 (ДМ с использованием кода Хемминга, нужно n-1 избыточных дисков), RAID 3 и 4 (ДМ с чередованием и выделенным диском четности), RAID 5 (ДМ с чередованием и "невыделенным диском четности"), RAID 6. Отличия RAID 3 от RAID 2: невозможность коррекции ошибок на лету и меньшая избыточность. RAID 4 похож на RAID 3, но отличается от него тем, что данные разбиваются на блоки, а не на байты. Таким образом, удалось отчасти «победить» проблему низкой скорости передачи данных небольшого объема. Основным недостатком уровней RAID 2-4 является невозможность производить параллельные операции

записи, так как для хранения информации о четности используется отдельный контрольный диск. RAID 5 не имеет этого недостатка. RAID 6 похож на RAID 5, но под контрольные суммы выделяется ёмкость 2-х дисков, рассчитываются 2 суммы по разным алгоритмам.

126. **Типы файлов устройств** - файлы байт-ориентированных устройств (побайтный обмен данными), файлы блок-ориентированных устройств (обмен фиксированными блоками данных). Особенностью работы с блок-ориентированными устройствами является возможность организации буферизации при обмене.
127. **Таблица драйверов** - bdevsw – таблица драйверов блок-ориентированных устройств. cdevsw - таблица байт-ориентированных устройств. Выбор конкретной таблицы определяется типом файла устройства. Соответственно, поле старший номер определяет строку таблицы, с которой ассоциирован драйвер устройства. Каждая запись этих таблиц содержит так называемый коммутатор устройства.
128. **Коммутатор устройства** – структура, в которой размещены указатели на соответствующие точки входа (функции) драйвера.
129. **Точки входа в драйвер** – bopen, bclose, bread, bwrite, bioctl (управление устройством, задание режимов работы драйвера), bintr (обработка прерываний), bstrategy (управление стратегией организации блок-ориентированного обмена)
130. **Таблица индексных дескрипторов открытых файлов** – содержит копию индексного дескриптора открытого файла и кратность - счетчик открытых в системе файлов, связанных с данным ИД. Данная таблица размещается в памяти ядра ОС. Вся работа с содержимым открытых файлов происходит посредством использования копии ИД, размещенной в таблице ТИДОФ.
131. **Таблица файлов** - каждая запись ТФ соответствует используемому файловому дескриптору, содержит указатели чтения/записи из/в файл. Если файловый дескриптор в процессе образуется за счет наследования, то в этом случае новые записи в ТФ не образуются, а происходит увеличение счетчика «наследственности» в записи, соответствующей файлу, открытому в прародителе. Таблица размещается в памяти ОС.
132. **Таблица открытых файлов** – связана с конкретным процессом. Номер записи в данной таблице есть номер ФД, который может использоваться в процессе. Каждая строка этой таблицы имеет ссылку на соответствующую строку ТФ. Первые три строки этой таблицы используются для файловых дескрипторов стандартных устройств/файлов ввода вывода.
133. **Стратегии управления ОП** - Одиночное непрерывное распределение (только один процесс, ОП делится на 2 области – область ОС и область процесса), Распределение разделами (память, не принадлежащую ОС, делим на конкретное количество разделов, каждом может быть свое задание и свой процесс. Первый алгоритм - статическое определение разделов, когда с каждым разделом связана своя очередь процессов. Второй алгоритм – динамическое распределение, когда одна очередь, и каждый раз осуществляется поиск наилучшего процесса для размещения), Распределение перемещаемыми разделами (фиксированное количество разделов, когда начинается внешняя фрагментация, происходит перемещение разделов и освобождение одного большого куска), Страничное распределение (содержимое таблицы страниц определяет соответствие виртуальной памяти физической для выполняющейся в данный момент программы/процесса), Сегментное распределение (виртуальное адресное пространство представляется в виде совокупности сегментов различной длины, каждый сегмент имеет свою виртуальную адресацию), Сегменто-страничное распределение.
134. **Файловая система** - часть операционной системы, представляющая собой совокупность организованных наборов данных, хранящихся на внешних запоминающих устройствах, и программных средств, гарантирующих именованный доступ к этим данным и их защиту.
135. **Модель ФС версии** System V – структура - [суперблок, область индексных дескрипторов, блоки файлов]. Файл-каталог для ФС System V представляет собой таблицу, каждая запись которой состоит из 16 байтов. Первые 2 байта – это номер индексного дескриптора, остальные–поле для имени файла, первые две строчки запезервированы. Поддержка жестких и символьных ссылок. Недостатки: Концентрация важной информации в суперблоке, Много ненадежных ссылочных структур, Фрагментация файла по диску, Короткие имена файлов.
136. **Модель ФС версии** FFS UNIX BSD - дисковое пространство имеет суперблок ,оно разделено на области одинакового размера, -группы цилиндров. ФС старается разместить блоки файлов в пределах одной группы цилиндров, стараясь располагать файлы в той же группе, что и каталог в котором они расположены. Структура группы циллиндров – резервная копия суперблока,

информация о свободных блоках и о свободных ИД, массив ИД, блоки файлов. Блоки разбиваются на фрагменты. Блок может использоваться для нескольких файлов только при хранении их последних байт, не занимающих всех фрагментов полного блока. Структура каждой записи каталога FFS - номер индексного дескриптора, длина записи в каталоге, длина имени файла, имя файла (дополненное до 256 символов).

137. **Файловый дескриптор** – системная структура данных, содержащая информацию о актуальном состоянии «открытого» файла.
138. **Каталог** – компонент файловой системы, содержащий информацию о содержащихся в файловой системе файлах.
139. **Суперблок** – содержит оперативную информацию о текущем состоянии файловой системы, а также данные о параметрах настройки, в частности: размер логического блока, размер файловой системы в логических блоках (включая суперблок), максимальное количество индексных дескрипторов, число свободных блоков, число свободных индексных дескрипторов, массив номеров свободных блоков, массив номеров свободных индексных дескрипторов.
140. **Индексный дескриптор** – описатель файла, содержит все необходимые для работы с файлом служебные атрибуты. Через ИД осуществляется доступ к содержимому файлов. Любое имя файла в системе ассоциировано с единственным ИД, но одному ИД может соответствовать произвольное количество имен. Главное, он содержит массив номеров блоков файла.
141. **Массив номеров свободных блоков** - все свободные блоки ФС организованы в однонаправленный список, структурная организация которого следующая: 1-й элемент этого списка – массив из N ссылок, которые размещаются в суперблоке. 0-й элемент этого массива есть номер блока из пространства блоков ФС, в котором находится продолжение этого массива. Соответственно 0-й элемент этого блока есть ссылка на следующий массив из N ссылок и т.д.
142. **Массив номеров свободных индексных дескрипторов** - содержит оперативный набор номеров свободных индексных дескрипторов. При освобождении индексного дескриптора, если для него нет свободного места в массиве, то этот номер «забывается». При запросе нового индексного дескриптора осуществляется поиск в массиве, если массив пустой – происходит операция обновления его содержимого (происходит просмотр области индексных дескрипторов и занесение в массив обнаруженных свободных).
143. **Адресация блоков файла** - массив номеров блоков файла В ИД содержит список из 13 номеров блоков на диске. Первые десять указывают на десять блоков некоторого файла. Если файл занимает более 10 блоков, то 11 элемент указывает на косвенный блок, содержащий до 128 адресов дополнительных блоков файла. Большие файлы используют 12-ый элемент, который указывает на блок, содержащий 128 указателей на блоки, каждый из которых содержит по 128 адресов блоков файла. Еще в больших файлах аналогично используется 13 элемент. Формула максимальной длины файла: (10 + A + A^2 + A^3)*[размер блока], где A = [размер блока]/[размер одного адреса (напр. int)].
144. **Содержание ИД файла устройства** - тип файла устройства (байториентированный или блокориентированный), «старший номер» устройства (номер драйвера в соответствующей таблице драйверов устройств), «младший номер» устройства (служебная информация, передающаяся драйверу устройства).
145. **Блок** – порция данных, фиксированного размера, в рамках которого идет обмен данными с устройством.
146. **Блоки файлов** - пространство на системном устройстве, в котором размещается вся информация, хранящаяся в файлах и о файлах, которая не поместилась в предыдущие блоки файловой системы.
147. **Файл** Unix – это специальным образом именованный набор данных, размещенный в файловой системе. Типы файлов UNIX: обычный файл, каталог, файл устройств, именованный канал, ссылка, сокет. Содержимое файла устройства целиком хранится в индексном дескрипторе. В поздних версиях UNIX содержимое файла-ссылки тоже хранится в индексном дескрипторе.
148. **Атрибуты файла** – имя, права доступа, персонификация (создатель, владелец), тип файла, размер записи, размер файла, указатель чтения / записи, время создания, время последней модификации, время последнего обращения, предельный размер файла.
149. **Жесткая связь** - с одним и тем же индексным дескриптором будет ассоциироваться два или более имени, размещенных в произвольных точках ФС. При этом каждое из этих имен равноценно.
150. **Символическая связь** - косвенная адресация на существующее имя файла.
151. **Квотирование пространства файловой системы** – Учет числа файлов: Гибкий лимит числа файлов, Жесткий лимит числа файлов, Количество файлов, Счетчик предупреждений, Учет числа блоков:

Гибкий лимит числа блоков, Жесткий лимит числа блоков, Количество блоков, Счетчик предупреждений.

152. **Стратегии архивирования** - Физическая архивация (копирование всех блоков, либо только использованных блоков), Логическая архивация (копирование файлов (а не блоков), модифицированных после заданной даты).
153. **Инкрементное архивирование** - архивирование, при котором в первый раз делается мастер-копия, после чего при каждой новой архивации к ней добавляются копии только тех файлов, которые были изменены или созданы с момента последней архивации.
154. **Контроль целостности файловой системы** – формируются две таблицы: таблица занятых блоков и таблица свободных блоков. Можно анализировать: либо они дополняют друг друга до всей ФС, тогда все в порядке, либо есть пропавший блок, либо какой-то блок дважды посчитан свободным, либо какой-то занятый блок дважды посчитан занятым.
155. **Полновесные процессы** - это процессы, выполняющиеся внутри защищенных участков памяти операционной системы, то есть имеющие собственные виртуальные адресные пространства для статических и динамических данных.
156. **Легковесные процессы** (**нити** или **сопрограммы**) - не имеют собственных защищенных областей памяти. Они работают в мультипрограммном режиме одновременно с активировавшей их задачей и используют ее виртуальное адресное пространство, в котором им при создании выделяется участок памяти под динамические данные (стек).
157. MPI - Message Passing Interface - программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.
158. **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 пользователя, владельца и т.п.).
159. **Процесс в ОС** Unix – объект, зарегистрированный в таблице процессов Unix.
160. **Процесс в ОС** Unix –объект, порожденный системным вызовом fork().
161. **Таблица процессов** – содержит pid, часть контекста, ссылки на остальные части контекста.
162. **Порождение процесса** – ~~сыновний~~ дочерний процесс наследует от родительского процесса окружение, открытые файлы (кроме тех, при открытии которых было указано не наследовать), способы обработки сигналов, разрешение переустановки эффективного идентификатора пользователя, разделяемые ресурсы процесса-отца, текущий рабочий каталог и домашний каталоги и т.д.
163. **Замена тела процесса** - изменяются следующие атрибуты процесса: тело процесса, режимы обработки сигналов (на по-умолчанию), если для новой программы установлен s –бит то изменяются эффективные идентификаторы владельца и группы, закрываются файлы для которых установлен флаг close-on-exec.
164. **Завершение 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.
165. **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.
166. **Основные задачи планирования** - Планирование очереди процессов на начало обработки, Планирование распределения времени ЦП между процессами, Планирование свопинга, Планирование обработки прерываний, Планирование очереди запросов на обмен.
167. **Приоритет процесса** – числовое значение, показывающее степень привилегированности процесса при использовании ресурсов ВС (в частности, времени ЦП).
168. **Невытесняющая стратегия планирования времени ЦП** - величина кванта не ограничена , никто принудительно не скидывает процесс с ЦП. Разработчики берут на себя функции диспетчера.

169. **Вытесняющая стратегия** - величина кванта ограничена.
170. HPF - highest priority first - планирование по наивысшему приоритету . При появлении в очереди готовых процессов процесса с более высоким приоритетом, чем у текущего наступает момент смены процесса. Смена происходит либо в тот же момент, когда приоритет произвольного процесса стал больше чем приоритет считающегося, либо после того, когда закончится квант времени считающегося.
171. **Группы приоритетов** UNIX (в порядке убывания) - Программа свопинга, Управление блочными устройствами I/O, Управление файлами, Управление байт-ориентированными устройствами I/O, Пользовательские процессы.
172. **Область свопинга** - специально выделенное системой пространство внешней памяти.
173. **Параллельные процессы** - процессы, выполнение которых хотя бы частично перекрывается по времени.
174. **Независимые процессы** – процессы, использующие независимое множество ресурсов и на результат работы такого процесса не влияет работа независимого от него процесса.
175. **Взаимодействующие процессы** - совместно используют ресурсы, и выполнение одного может оказывать влияние на результат другого.
176. **Разделение ресурса** - совместное использование несколькими процессами ресурса ВС, когда каждый из процессов одновременно владеет ресурсом.
177. **Критические ресурсы** - разделяемые ресурсы, которые должны быть доступны в текущий момент времени только одному процессу.
178. **Состояние гонки** (race condition) — ошибка проектирования многозадачной системы, при которой работа системы зависит от того, в каком порядке выполняются части кода.
179. **Критическая секция** - часть программы (фактически набор операций), в которой осуществляется работа с критическим ресурсом.
180. **Взаимное исключение** – способ работы с разделяемым ресурсом, при котором постулируется, что в тот момент, когда один из процессов работает с разделяемым ресурсом, все остальные процессы не могут иметь к нему доступ.
181. **Блокирование** (**дискриминация**) – это ситуация, когда один из процессов никогда не получит доступа к ресурсу, т.е. будет ожидать вечно.
182. **Тупик** (deadlock) – ситуация, в которой конкурирующие за критический ресурс процессы вступают в клинч – безвозвратно блокируются.
183. **Способы реализации взаимного исключения** - Запрещение прерываний и специальные инструкции, Алгоритм Петерсона, Активное ожидание, Семафоры Дейкстры, Мониторы, Обмен сообщениями.
184. **Семафоры Дейкстры** – объект (переменная типа семафор), над которой определены две атомарные операции: down и up. Down проверяет значение семафора, и если оно больше нуля, то уменьшает его на 1. Если же это не так, процесс блокируется, причем операция down считается незавершенной. Up увеличивает значение семафора на 1. При этом, если в системе присутствуют процессы, блокированные ранее при выполнении down на этом семафоре, ОС разблокирует один из них с тем, чтобы он завершил выполнение операции down.
185. **Монитор** – языковая конструкция, представляет собой совокупность процедур и структур данных, объединенных в программный модуль специального типа. Три основных свойства: 1. Структуры данных, входящие в монитор, могут быть доступны только для процедур, входящих в этот монитор. 2. Процесс «входит» в монитор путем вызова одной из его процедур. 3. В любой момент времени внутри монитора может находиться не более одного процесса. Если процесс пытается попасть в монитор, в котором уже находится другой процесс, он блокируется.
186. **Обмен сообщениями** – средство, которое может быть использовано как для синхронизации, в частности для организации взаимного исключения, так и для обмена информацией между взаимосвязанными процессами, выполняющими общую работу. две операции: send и receive. Они могут быть блокирующими (send блокируется, пока сообщение не будет доставлено, receive блокируется, пока сообщение не будет получено) и не блокирующими.
187. **Задача «Обедающие философы»** - 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.

188. **Задача «Читатели и писатели» -** 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.
189. **Задача о «спящем парикмахере» -** 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.
190. **Сигнал** – средство уведомления процесса о наступлении некоторого события в системе.
191. **Неименованный канал -** некая сущность, в которую можно помещать и извлекать данные по принципу FIFO, для чего служат два файловых дескриптора, ассоциированных с каналом: один для записи в канал (write(fd[1],...)), другой — для чтения (read(fd[0],...)). Для создания канала служит системный вызов pipe. Главное отличие от файла – это то, что в нем реализуется строго последовательный доступ к данным. При отсутствии переполнения, UNIX гарантирует атомарность записи и чтения из канала.
192. **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.
193. **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.
194. 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.
195. **Трассировка** — процесс пошагового выполнения программы.
196. **Межпроцессное взаимодействие** (Inter-Process Communication, IPC) — набор способов обмена данными между процессами. Состав: Очереди сообщений, Семафоры, Разделяемая память.
197. **Очередь сообщений** IPC - 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.
198. **Разделяемая память** IPC - 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.
199. **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, Датаграммное соединение.

Составлено на основании конспекта Светланы Граур.
Serious, 2011.