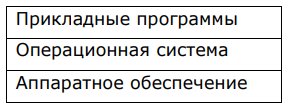
**Вопросы к экзамену**

1. **Операционные системы: определение, назначение, состав, функции.**

Операционная система (ОС) — это комплекс программ, который служит связующим звеном между аппаратными ресурсами компьютера и прикладными программами.

 Она управляет ресурсами компьютера, обеспечивает их эффективное использование и организует взаимодействие с пользователем. ОС предоставляет прикладным программам удобный интерфейс для работы с аппаратными ресурсами, скрывая от них сложность управления этими ресурсами. Конечные пользователи взаимодействуют с ОС через пользовательский интерфейс, который может быть командной строкой или графическим интерфейсом.

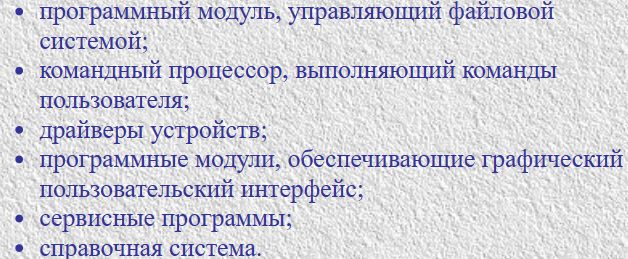
Назначение

Это программы, которые управляют ресурсами компьютера и облегчают взаимодействие пользователя с компьютером.

Она выполняет две основные функции:

* предоставляет удобный интерфейс для работы с компьютером;
* управляет ресурсами компьютера.

Состав (подсистемы):

* управления процессами,
* управления памятью,
* управления файлами,
* управления внешними устройствами,
* пользовательского интерфейса,
* защиты данных и администрирования.

Функции:

Основные

1. **Управляет устройствами компьютера**: ОС координирует работу всех аппаратных средств компьютера, таких как процессор, оперативная память, диски и т.д.
2. **Управляет процессами**: ОС контролирует выполнение программ и их взаимодействие с аппаратными устройствами.
3. **Управляет доступом к данным**: ОС контролирует доступ к данным, хранящимся на различных носителях, таких как жесткий диск или компакт-диск, обычно с помощью файловой системы.
4. **Ведет файловую структуру**: ОС организует файлы на компьютере в структурированном виде.
5. **Предоставляет пользовательский интерфейс**: ОС обеспечивает диалог с пользователем, позволяя ему взаимодействовать с компьютером.

Функции: Перефразированные из лк:

1. **Прием заданий**: ОС принимает команды от пользователя и обрабатывает их.
2. **Загрузка программ**: ОС загружает программы в оперативную память для выполнения.
3. **Управление задачами**: ОС управляет запуском, приостановкой и остановкой программ. Она также координирует взаимодействие между различными задачами.
4. **Управление файлами и базами данных**: ОС обеспечивает работу систем управления файлами и базами данных.
5. **Многозадачность**: ОС позволяет одновременно выполнять несколько задач.
6. **Планирование задач**: ОС определяет, какие задачи должны быть выполнены и в каком порядке.
7. **Управление вводом/выводом**: ОС управляет операциями ввода и вывода, такими как чтение и запись на диск.
8. **Управление памятью**: ОС управляет использованием оперативной и виртуальной памяти.
9. **Обмен сообщениями**: ОС обеспечивает обмен данными между программами.
10. **Защита данных**: ОС защитит одну программу от влияния другой и обеспечит сохранность данных.
11. **Безопасность**: ОС обеспечивает аутентификацию, авторизацию и другие меры безопасности.
12. **Обработка сбоев**: ОС предоставляет услуги для восстановления после частичных сбоев системы.
13. **Поддержка программирования**: ОС обеспечивает работу систем программирования.
14. **Операционные системы: классификация, основные этапы развития, особенности современного этапа развития**

Классификации:

1. **По алгоритму управления процессором**:
   * Однозадачные: выполняют одну задачу за раз.
   * Многозадачные: могут выполнять несколько задач одновременно.
2. **По количеству пользователей**:
   * Однопользовательские: предназначены для одного пользователя.
   * Многопользовательские: позволяют одновременную работу нескольких пользователей.
3. **По типу устройства**:
   * Локальные: устанавливаются на одно устройство.
   * Сетевые: обеспечивают работу в сети.
4. **По количеству процессоров**:
   * Однопроцессорные: работают с одним процессором.
   * Многопроцессорные: могут работать с несколькими процессорами одновременно.
5. **Роль в сети:** 
   * серверные (управляют ресурсами сети),
   * клиентские (получают доступ к ресурсам сети).
6. **Тип лицензии:** 
   * открытые (с открытым исходным кодом, можно изучать и редактировать),
   * проприетарные (с закрытым исходным кодом).
7. **Сфера использования:** 
   * ОС мэйнфреймов – больших компьютеров; серверов, персональных компьютеров, мобильных устройств, встроенные ОС, ОС маршрутизаторов**.**

Развитие операционных систем (ОС) прошло через несколько ключевых этапов:

1. **Первое поколение (1945-1955 гг.)**: ОС отсутствовали. Компьютеры работали на электронных лампах и были примитивными. [Программы писались в машинных кодах и загружались вручную1](https://scienceforum.ru/2020/article/2018018548).
2. **Второе поколение (1955-1965 гг.)**: Компьютеры начали использовать **транзисторы**, что увеличило их надежность и время работы. [В это время появились первые системные программы, такие как **компилятор** Fortran и ассемблер для IBM-7011](https://scienceforum.ru/2020/article/2018018548).
3. [**Третье поколение операционных систем (1965-1980 гг.)** характеризовалось использованием **интегральных схем**, что улучшило качество и снизило стоимость компьютеров1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC). [В это время были реализованы ключевые концепции, присущие современным ОС, такие как разделение времени и **многозадачность**, разделение полномочий, реальный масштаб времени, файловые структуры и файловые системы1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC)[2](https://bing.com/search?q=%D0%A2%D1%80%D0%B5%D1%82%D1%8C%D0%B5+%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5+%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85+%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC). [Примерами операционных систем третьего поколения являются OS/360, CTSS, MULTICS1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC). [Упрощенная однопользовательская версия системы MULTICS позже стала операционной системой UNIX®, которая стала популярной в академических кругах, правительственных учреждениях и во многих компаниях1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC).
4. [**Четвертое поколение операционных систем (1980-2005 гг.)** было связано с развитием **больших интегральных схем**, что привело к появлению микрокомпьютеров или **персональных компьютеров**1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC)[2](https://bing.com/search?q=%D0%A7%D0%B5%D1%82%D0%B2%D0%B5%D1%80%D1%82%D0%BE%D0%B5+%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5+%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85+%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC). [Операционные системы этого поколения, такие как CP/M и MS-DOS, были разработаны для этих микропроцессоров1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC)[2](https://bing.com/search?q=%D0%A7%D0%B5%D1%82%D0%B2%D0%B5%D1%80%D1%82%D0%BE%D0%B5+%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5+%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85+%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC). [С течением времени, по мере увеличения сложности задач, возникла необходимость в более сложных операционных системах, что привело к возрождению многих функций, характерных для больших вычислительных систем1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC)[2](https://bing.com/search?q=%D0%A7%D0%B5%D1%82%D0%B2%D0%B5%D1%80%D1%82%D0%BE%D0%B5+%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5+%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85+%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC). [В результате появились операционные системы с графическим интерфейсом, такие как **Mac OS X, Windows и UNIX**-подобные системы1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC)[2](https://bing.com/search?q=%D0%A7%D0%B5%D1%82%D0%B2%D0%B5%D1%80%D1%82%D0%BE%D0%B5+%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5+%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85+%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC). [Кроме того, развитие компьютерных сетей привело к появлению сетевых и распределенных операционных систем1](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC)
5. **Пятое поколение операционных систем (2005 г. - по н.в.)**

* [**Смартфоны**: Появление смартфонов, которые объединяют функции телефона и компьютера1](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D1%8B_%D0%BF%D1%8F%D1%82%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)[2](https://sariola.ru/all_of_about_pc/na-poroge-patogo-pokolenia-vycislitelnoj-tehniki-kakie-izmenenia-zdut-zavtrasnie-pk.html).
* **Многоядерные процессоры**: В 2005 году появились **многоядерные процессоры**, что потребовало создания многоядерных операционных систем. [Операционные системы, такие как Windows и Linux, были успешно адаптированы для работы на многопроцессорной (многоядерной) архитектуре1](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D1%8B_%D0%BF%D1%8F%D1%82%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)[2](https://sariola.ru/all_of_about_pc/na-poroge-patogo-pokolenia-vycislitelnoj-tehniki-kakie-izmenenia-zdut-zavtrasnie-pk.html).
* **Конкуренция мобильных ОС**: На рынке мобильных устройств возникла жесткая конкуренция между различными операционными системами. [На данный момент доминирующими являются Google Android и Apple iOS1](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D1%8B_%D0%BF%D1%8F%D1%82%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)[2](https://sariola.ru/all_of_about_pc/na-poroge-patogo-pokolenia-vycislitelnoj-tehniki-kakie-izmenenia-zdut-zavtrasnie-pk.html).
* **Виртуализация и облачные технологии**: Широкую популярность приобрела технология виртуализации, которая позволяет представлять вычислительные ресурсы, абстрагированные от аппаратной реализации. [Это привело к появлению облачных технологий, где потребности клиентов в вычислениях и хранении данных удовлетворяются специализированными дата-центрами](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D1%8B_%D0%BF%D1%8F%D1%82%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)

Современные операционные системы (ОС) развиваются в нескольких ключевых направлениях:

1. **Безопасность**: С учетом растущей ценности информации и угроз безопасности, современные ОС активно развивают и улучшают механизмы защиты, включая шифрование данных, аутентификацию и авторизацию.
2. **Многоплатформенность**: Современные ОС способны работать на различных типах компьютеров, что делает их более универсальными.
3. **Удобство использования**: Удобство работы пользователя с компьютером становится все более важным, поэтому современные ОС стремятся упростить интерактивную работу с компьютером, включая развитие графических интерфейсов и интеллектуальных систем, которые могут направлять действия пользователя и принимать за него рутинные решения.
4. **Прозрачность сетевых ресурсов**: Сетевые ОС стремятся обеспечить высокий уровень прозрачности сетевых ресурсов, чтобы пользователи и администраторы сети могли легко определить местоположение ресурсов, а разработчики сетевых приложений могли легко определить местоположение данных и программных модулей в сети. Ожидается, что будущие ОС смогут организовать распределенные вычисления, превратив сеть в виртуальный компьютер.
5. Компоненты архитектуры вычислительных систем, их назначение и взаимодействие. (Понятие общей шины. Контроллеры. Компоненты компьютера: процессор, память и др) (risc , c)

Классическая архитектура ЭВМ

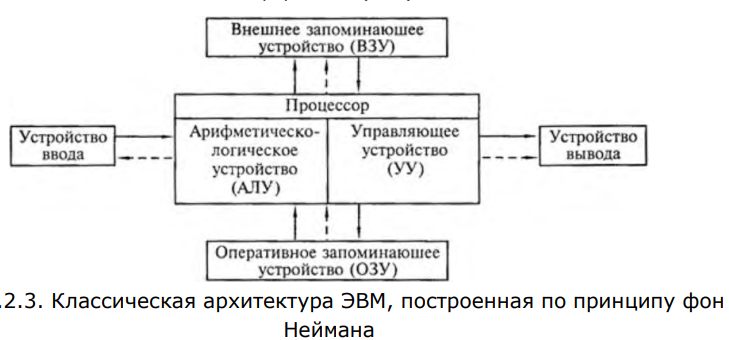
• арифметическо-логическое устройство (АЛУ), выполняющее арифметические и логические операции;

• управляющее устройство (УУ), организующее процесс выполнения программ;

• внешнее запоминающее устройство (ВЗУ), или память, для хранения программ и данных;

• оперативное запоминающее устройство (ОЗУ);

• устройства ввода и вывода информации (УВВ).



Конфигурация типичного современного компьютера включает следующие компоненты:

1. Системный блок:
   * Блок питания: Поставляет электроэнергию всем компонентам компьютера.
   * Корпус: Оболочка, которая содержит и защищает внутренние компоненты компьютера.
   * Материнская плата: Основная плата, на которой устанавливаются ЦП, ОЗУ, разъемы для подключения устройств и другие компоненты.
   * Процессор (ЦП): Выполняет вычислительные операции и управляет работой компьютера.
   * Жесткий диск: Хранит операционную систему, программы и файлы пользователя.
   * Оперативная память (ОЗУ): Используется для временного хранения данных и выполнения программ.
   * Видеоадаптер: Обрабатывает графическую информацию и отправляет ее на монитор для отображения.
   * Звуковой адаптер: Обрабатывает аудиоинформацию и управляет звуковым выводом.
   * Приводы дисков: Например, оптический привод (CD/DVD) или SSD-накопитель для чтения и записи данных на диски.
   * Шлейфы, кабели: Используются для подключения и связи устройств между собой и с материнской платой.
2. Монитор: Отображает информацию и графику компьютера пользователю.
3. Клавиатура: Позволяет пользователю вводить текст и команды в компьютер.
4. Мышь: Используется для перемещения указателя по экрану и взаимодействия с графическим интерфейсом.
5. Периферийные устройства: Включают в себя принтеры, сканеры, веб-камеры, акустические системы и другие устройства, которые расширяют функциональность компьютера.

Эти компоненты работают вместе, обеспечивая функционирование компьютерной системы. Например, процессор выполняет инструкции, хранящиеся в оперативной памяти, и получает данные из жесткого диска. Видеоадаптер обрабатывает графическую информацию и отправляет ее на монитор для отображения. Клавиатура и мышь позволяют пользователю взаимодействовать с компьютером. Периферийные устройства расширяют функциональность и возможности компьютера.



Любая шина(это соединение для передачи данных между функциональными блоками компьютера) состоит из трех частей:

• шина данных, по которой передается информация;

• шина адреса, определяющая, кому передаются данные;

• шина управления, регулирующая процесс обмена информацией

Для связи между компонентами компьютерной системы используется общая шина, которая состоит из нескольких проводников и служит для передачи данных, адресов и управляющих сигналов. Шины могут быть внутренними (соединяющими процессор и АЛУ) и внешними (соединяющими процессор с памятью и устройствами ввода-вывода). Каждый тип шины имеет свои особенности и требования.

Контроллеры(это устройство или программа для передачи данных между функциональными блоками компьютера\ Например, контроллер клавиатуры получает информацию о нажатии клавиши и передает ее в центральный процессор для дальнейшей обработки)  
  
 - специализированные устройства, которые управляют внешними устройствами и обеспечивают связь между ними и центральным процессором компьютера. Они выполняют определенные функции ввода-вывода и освобождают процессор от управления каждым устройством непосредственно. Контроллеры имеют свою систему команд и специализированные возможности для работы с конкретными устройствами. Их использование повышает эффективность работы системы.

4. Система прерываний. Аппаратная поддержка в процессорах x86. Различие работы в реальном и защищенном режимах процессора. Векторы прерываний. (первоначальный вопрос: Задачи операционной системы по управлению и организации работы компьютера.)

Система прерываний обеспечивающие временное прекращение выполнения последовательности команд для перехода к выполнению др. последовательности команд или для возвращения к ранее прерванной программе.

(аппаратные, программные, исключения, внешние, внутренние)

**Системный вызов**: Пользовательская программа делает системный вызов для получения услуг от ОС. Этот вызов перехватывается внутри ядра ОС и запускает соответствующую службу.

Реальный и защищенный режимы процессора отличаются по следующим основным параметрам:

1. **Реальный режим**: В этом режиме обращение к оперативной памяти происходит по реальным (действительным) адресам. [Набор доступных операций не ограничен, и защита памяти не используется1](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B6%D0%B8%D0%BC_%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0).
2. **Защищенный режим**: В защищенном режиме обращение к памяти происходит по виртуальным адресам с использованием механизмов защиты памяти. [Набор доступных операций определяется уровнем привилегий1](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B6%D0%B8%D0%BC_%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0).

Обработка прерывания в процессорах x86 состоит из трех этапов:

Обязательными для сохранения являются регистры **cs**, **ip**, **flags**

* 1. **Прекращение выполнения** текущей программы и **сохранение** ее состояния в стеке.
  2. Переход к выполнению программы-обработчика прерывания, выбранной по номеру источника прерывания из таблицы векторов прерываний.
  3. Возврат управления прерванной программе после восстановления ее состояния из стека.

**Векторы прерываний**: Все прерывания и особые ситуации имеют уникальные номера ID, называемые векторами прерываний. Они используются для идентификации и обработки различных типов прерываний. (0 до 31 отведены для особых ситуаций, от 32 до 255 свободны для любого использования)

5. Структура ядра и его функции. Объекты ядра. Основные операции над объектами ядра.

Наиболее общим подходом к структуризации ОС является подразделение модулей **две группы:**

• модули, выполняющие основные функции ОС — **ядро ОС;**

• модули, выполняющие вспомогательные функции ОС.

Модули ядра выполняют базовые функции ОС

• управление процессами;

• управление памятью;

• управление устройствами ввода-вывода.

Функции, входящие в состав ядра, можно разделить на два класса.

1. **Внутрисистемные функции**: Это функции, которые обеспечивают работу самой системы, такие как переключение между процессами, управление памятью и обработка прерываний. Эти функции не доступны для приложений.
2. **Функции поддержки приложений**: Это функции, которые доступны для приложений и обеспечивают им доступ к ресурсам системы. Они формируют интерфейс прикладного программирования (API), который используется приложениями для обращения к ядру с запросами или системными вызовами.

Ядро операционной системы (ОС) — это центральная часть ОС, которая обеспечивает приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации. Оно также предоставляет сервисы файловой системы и сетевых протоколов.

**Объекты ядра** — это коллекция данных, которыми управляет операционная система.

**Монолитное, Модульное, Микроядро, Экзоядро,**

Типичные объекты режима ядра включают следующие объекты:

1. Объекты устройств.
2. Объекты файлов.
3. Символические ссылки.
4. Разделы реестра.
5. Потоки и процессы.
6. • объекты диспетчера ядра, такие как объекты событий и объекты мьютексов;
7. • объекты обратного вызова;
8. • объекты section.

**Основные операции над объектами ядра** включают:

1. Создание объекта и дескриптора объекта.
2. Получение нового существующего дескриптора объекта.
3. Закрытие дескриптора объекта. Когда приложение закрывает последний дескриптор для объекта ядра, система удаляет объект из памяти.

6. Утилиты. Системные обрабатывающие программы. Библиотеки процедур. Программы дополнительных услуг.

Утилиты – Вспомогательное (или служебное) программное обеспечение.

Утилиты - это специализированные программы, которые помогают обеспечивать эффективное функционирование компьютерной системы. Они поддерживают работу операционной системы, выполняя различные задачи, включая обнаружение и удаление вирусов, резервное копирование данных, управление файлами и дисками, и многое другое.

Утилиты облегчают доступ к различным функциям и настройкам, которые могут быть сложными для изменения без специализированных инструментов. Они могут быть частью операционной системы, поставляться вместе с определенным оборудованием или распространяться отдельно.

Типы утилит:

1. **Антивирусы**: Программы, предназначенные для обнаружения и удаления вредоносного программного обеспечения. Примеры: McAfee Antivirus, Quickheal Antivirus, Windows Defender.
2. **Системы управления файлами**: Инструменты для управления и организации файлов в системе. Примеры: Проводник Windows, Google Desktop, Double Commander.
3. **Инструменты сжатия**: Программы, используемые для сжатия файлов для экономии пространства хранения. Примеры: WinZip, WinRAR, 7-Zip.
4. **Инструменты управления дисками**: Программы для управления данными на дисках, включая разделение устройств на разделы. Примеры: MiniTool Partition Wizard, Paragon Partition Manager.
5. **Инструменты очистки диска**: Программы для освобождения дискового пространства путем удаления неиспользуемых файлов. Примеры: Razer Cortex, Piriform CCleaner.
6. **Дефрагментация диска**: Инструменты для уменьшения фрагментации и ускорения доступа к данным путем переупорядочения файлов. Примеры: Perfect disk, Deflaggler.
7. **Утилита резервного копирования**: Программы для создания резервных копий файлов, папок, баз данных или дисков для восстановления данных в случае их потери.

Обрабатывающие системные программы выполняют две основные функции:

**перенос информации** и **её преобразование**. Они делятся на **утилиты**, основная функция которых - перенос информации, и **лингвистические процессоры**, основная функция которых - перевод алгоритма с одного языка на другой.

Лингвистические процессоры включают **трансляторы (***компиляторы* и *ассемблеры***)** и **интерпретаторы**.

Трансляторы преобразуют алгоритм, записанный на языке программирования, в алгоритм, записанный на машинном языке. Они делятся на *компиляторы* и *ассемблеры*. Компиляторы используются для языков высокого уровня, таких как Паскаль и Си, а ассемблеры - для языка ассемблера.

Интерпретаторы и трансляторы - это обрабатывающие системные программы. Интерпретаторы переводят и выполняют каждый оператор исходной программы по очереди, в то время как трансляторы преобразуют всю программу целиком.

Примером интерпретатора является командный процессор, такой как cmd.exe или Powershell в Windows, sh или bash в Linux, и кроссплатформенный PowerShell (pwsh).

Командный файл - это текстовый файл с набором команд, которые последовательно выполняются командным процессором.

Наиболее часто используемые системные обрабатывающие программы включают текстовые и графические редакторы, компиляторы, компоновщики и отладчики.

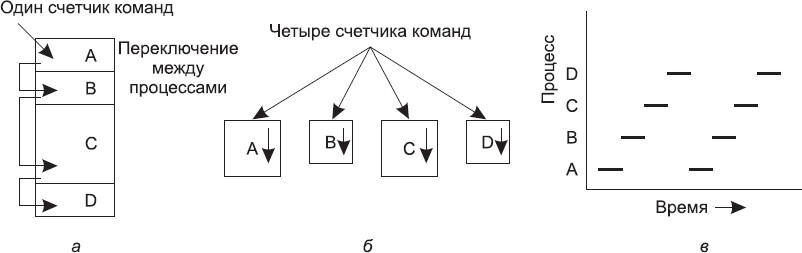
Библиотеки - это наборы функций, которые используются в различных программах для выполнения определенных задач. Они могут быть **статическими** (встроенными в программу) или **динамическими** (загружаются в оперативную память при необходимости).

В Windows системные библиотеки хранятся в папках System32 (для 64-битных библиотек) и SysWOW64 (для 32-битных библиотек).

В Linux библиотеки обычно находятся в папках /lib и /usr/lib. Путь к библиотекам указывается в файле /etc/ld.so.conf.

Операционные системы могут включать различные программы дополнительных услуг, такие как встроенные браузеры, почтовые клиенты, медиаплееры, графические редакторы, антивирусное программное обеспечение и утилиты системы. Эти программы предоставляют возможности для просмотра веб-страниц, отправки и получения электронной почты, воспроизведения аудио и видео, редактирования изображений, защиты от вредоносных программ и управления системными ресурсами.

**7. Понятие процесса. Системные и пользовательские процессы. Операции над процессами.**

**Процесс** — это просто экземпляр выполняемой программы, включая текущие значения счетчика команд, регистров и переменных. Концептуально у каждого процесса есть свой, виртуальный, центральный процессор. Постоянное переключение между процессами называется **мультипрограммированием**, или **многозадачным** режимом работы.

Компьютер: а — четыре программы, работающие в многозадачном режиме; б — концептуальная модель четырех независимых друг от друга

последовательных процессов; в — в отдельно взятый момент активна только одна программа

Процессы, которые выполняют системный код, называются **системными** и применяются к системе в целом. Они занимаются выполнением таких служебных задач, как распределение памяти, обмен страницами между внутренним и вспомогательным запоминающими устройствами, контроль устройств и т.п.

**Пользовательские** процессы выполняют собственный код и иногда обращаются к системным функциям. Выполняя собственный код, пользовательский процесс пребывает в пользовательском режиме (user mode).

**Операции над процессами.**

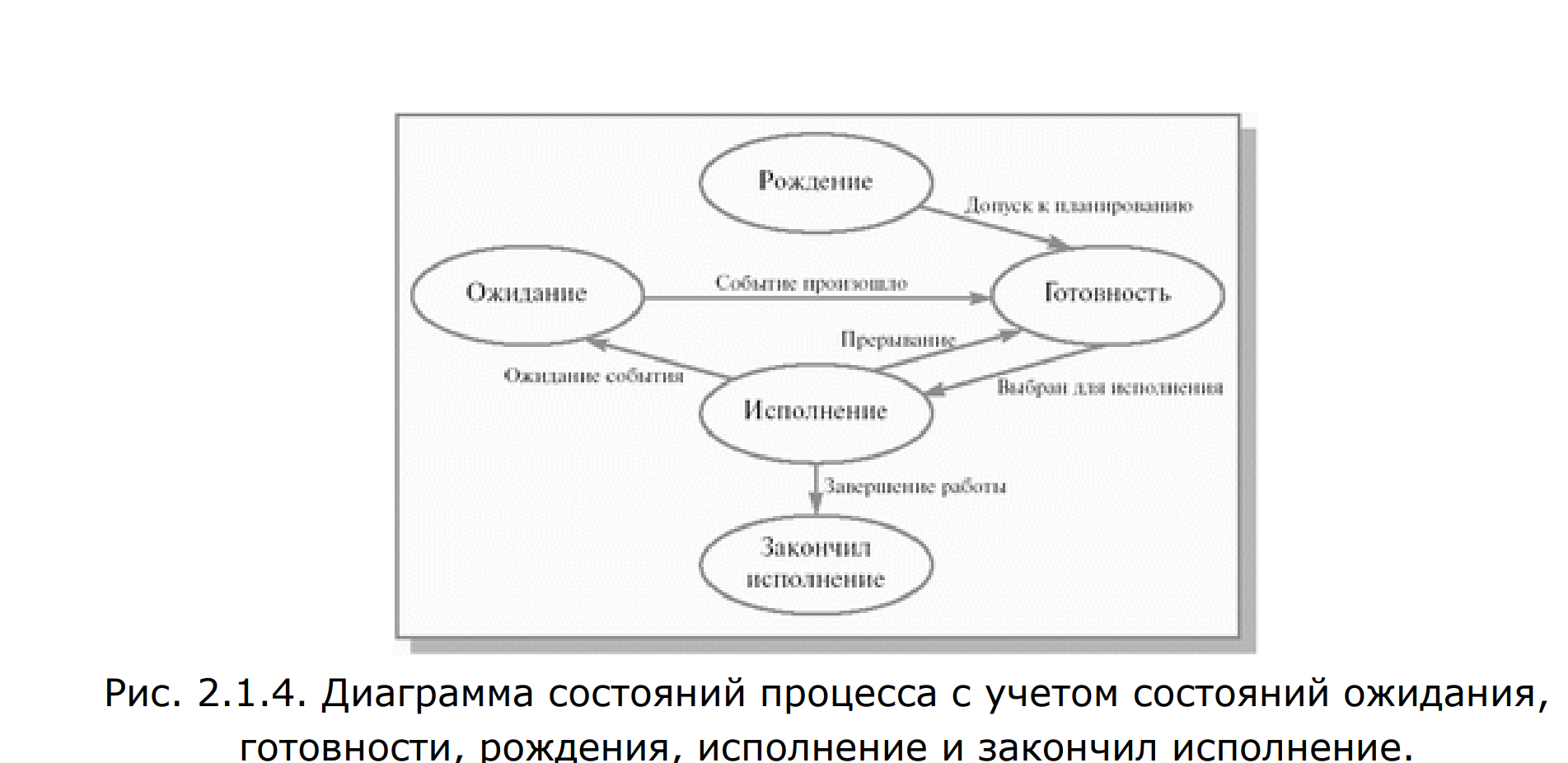
**Запуск процесса**. Из числа процессов, находящихся в состоянии готовность, операционная система выбирает один процесс для последующего исполнения.

**Приостановка процесса**. Работа процесса, находящегося в состоянии исполнение, приостанавливается в результате какого-либо прерывания.

**Блокирование процесса**. Процесс блокируется, когда он не может продолжать работу, не дождавшись возникновения какого-либо события в вычислительной системе.

**Разблокирование процесса**. После возникновения в системе какого-либо события операционной системе нужно точно определить, какое именно событие произошло

**Переключение контекста**

В действительности же деятельность мультипрограммной операционной системы состоит из цепочек операций, выполняемых над различными процессами, и сопровождается переключением процессора с одного процесса на другой.  


**8. Организация межпроцессного взаимодействия в ОС. Сигналы. Каналы. Классические проблемы межпроцессного взаимодействия.**

При выполнении параллельных процессов может возникать проблема, когда каждый процесс, обращающийся к разделяемым данным, исключает для всех других процессов возможность одновременного с ним обращения к этим данным – это называется **взаимоисключением.** Ресурс, который допускает обслуживание только одного пользователя за один раз, называется критическим ресурсом. Для организации коммуникации между одновременно работающими процессами применяются средства межпроцессного взаимодействия (Interprocess Communication - IPC).

**Средства локального уровня** IPC привязаны к процессору и возможны только в пределах компьютера.

**Удаленные IPC** предоставляют механизмы, которые обеспечивают взаимодействие как в пределах одного процессора, так и между программами на различных процессорах, соединенных через сеть.

Под **высокоуровневыми IPC** обычно подразумеваются пакеты программного обеспечения, которые реализуют промежуточный слой между системной платформой и приложением.

**Сигнал** в операционных системах семейства Unix — асинхронное уведомление процесса о каком-либо событии, один из основных способов взаимодействия между процессами. Отдельные сигналы подразделяются на три класса:

* системные сигналы (ошибка аппаратуры, системная ошибка и т.д.);
* сигналы от устройств;
* сигналы, определенные пользователем.

**Канал** (pipe) представляет собой средство связи стандартного вывода одного процесса со стандартным вводом другого. После создания канала, процесс может при помощи обычного системного вызова **write**() выводить данные в него, а затем вводить их, вызывая соответственно функцию **read**(). При выполнении вызова **fork**() дескрипторы канала наследуются процессом-"потомком". Таким образом, оба процесса получают возможность обмениваться данными.

**Проблемы:**

**Синхронный доступ**. Если все процессы считывают данные из файла, то в большинстве случае проблем не возникает. Однако, при попытке одним из процессов изменить этот файл, могут возникнуть так называемые конкурентные ситуации (race conditions).

**Дисциплина доступа**. Если нужно, чтобы как можно большее количество пользователей могли записывать данные, организуется так называемая очередь (по правилу «один пишет, несколько читают»). Практически организуется две очереди: одна — для чтения, другая — для записи. Такую дисциплину доступа можно организовать с помощью **барьеров** (блокировок). При этом создается общий барьер для считывателей, так как несколько процессов могут одновременно считывать данные, а также отдельный барьер для процесса-писателя. Такая организация предотвращает взаимные помехи в работе.

**Голодание процессов**. Организация дисциплины доступа может привести к ситуации, когда процесс будет длительно ждать своей очереди для записи данных. Поэтому иногда нужно организовывать очереди с приоритетами.

Если нельзя точно определить, какой из процессов запрашивает или возвращает свои данные в нужный компьютер первым, используется так называемое взаимодействие по модели "клиент-сервер". При этом используются один или несколько клиентов и один сервер. Клиент посылает запрос серверу, а сервер отвечает на него. После этого клиент должен дождаться ответа сервера, чтобы продолжать дальнейшее взаимодействие. Такое поведение называется **управлением потоком**. При одновременном доступе здесь также могут использоваться очереди с приоритетами.

Классический **тупик** возникает, если процесс A получает доступ к файлу A и ждет освобождения файла B. Одновременно процесс B, получив доступ к файлу B, ждет освобождения файла A. Оба процесса теперь ждут освобождения ресурсов другого процесса и не освобождают при этом собственный файл.

**9. Концепция потока. Параллельное исполнение потоков. Главный поток процесса.**

**Поток выполнения** (англ. thread — нить) — наименьшая единица обработки, исполнение которой может быть назначено ядром операционной системы. Несколько потоков выполнения могут существовать в рамках одного и того же процесса и совместно использовать ресурсы, такие как память, тогда как процессы не разделяют этих ресурсов. В частности, потоки выполнения разделяют инструкции процесса (его код) и его контекст (значения переменных, которые они имеют в любой момент времени).

Термин "параллельность" рассматривается в контексте вытесняющей многозадачности операционной системы - то есть, операционная система выделяет потоку некоторый квант времени, а затем переключается на другой поток.

**Многопоточность**: процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть **без предписанного порядка во времени**.

По умолчанию процесс создается с одним потоком, называемым **главным** или основным потоком. В Linux потоки реализованы с помощью системного вызова **clone**(), который как минимум в 10 раз меньше занимает времени для создания еще одного потока, чем создание еще одного процесса при помощи **fork**(). Такая скорость достигается за счет того, что многие атрибуты процесса разделяются между потоками.

В Windows процесс может породить практически неограниченное количество потоков. Для этого используется функция **CreateThread**. Процесс будет активен, пока активен хотя бы один поток.

**10. Диаграммы состояния потоков. Понятие контекста и переключения контекста.**

Для каждого созданного потока в системе предусматриваются три возможных его состояния:

• состояние выполнения, когда код потока выполняется процессором; на однопроцессорных платформах в этом состоянии в каждый момент времени может находиться только один поток;

• состояние готовности к выполнению, когда поток готов продолжать свою работу и ждет освобождения ЦП;

• состояние ожидания наступления некоторого события; в этом случае поток не претендует на время ЦП, пока не наступит определенное событие (завершение операции ввода/вывода, освобождение необходимого потоку занятого ресурса, сигнала от другого потока), часто такие потоки называют блокированными. Изменение состояния потока происходит в результате соответствующих действий. Удобно для этих целей использовать следующую диаграмму состояний и переходов (рис. 2.2.5)



**Переключение контекста** — в многозадачных ОС и средах — процесс прекращения выполнения процессором одной задачи (процесса, потока, нити) с сохранением всей необходимой информации и состояния, необходимых для последующего продолжения с прерванного места, и восстановления и загрузки состояния задачи, к выполнению которой переходит процессор. В процедуру переключения контекста входит так называемое планирование задачи — процесс принятия решения, какой задаче передать управление. При переключении контекста происходит сохранение и восстановление следующей информации:

• Регистровый контекст регистров общего назначения (в том числе флаговый регистр)

• Контекст состояния сопроцессора с плавающей точкой / регистров MMX (x86)

• Состояние регистров SSE, AVX (x86)

• Состояние сегментных регистров (x86)

• Состояние некоторых управляющих регистров (например, регистр CR3, отвечающий за страничное отображение памяти процесса) (x86)

**11. Многозадачность в ОС. Типы многозадачности.**

Многозадачность — свойство операционной системы или среды выполнения обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких задач. Истинная многозадачность операционной системы возможна только в распределённых вычислительных системах.

Существует два типа многозадачности:

• Процессная многозадачность (основанная на процессах — одновременно выполняющихся программах). Здесь программа — наименьший элемент управляемого кода, которым может управлять планировщик операционной системы. Более известна большинству пользователей (работа в текстовом редакторе и прослушивание музыки).

• Поточная многозадачность (основанная на потоках). Наименьший элемент управляемого кода — поток (одна программа может выполнять 2 и более задачи одновременно)

К псевдопараллельной многозадачности можно отвести следующие типы: простое переключение, совместная (или кооперативная) многозадачность и вытесняющая (или приоритетная) многозадачность.

1. \*\*Простое переключение:\*\*

- Загружает в память два или более приложения, но предоставляет процессорное время только основному.

- Преимущества: Может использовать уже работающие программы без учета многозадачности. Недостатки: Ограниченное взаимодействие между программами.

2. \*\*Совместная многозадачность:\*\*

- Задача выполняется после объявления готовности отдать процессорное время другим задачам.

- Преимущества: Упрощает программирование, отсутствие необходимости в защите данных. Недостатки: Затруднена реализация многозадачной архитектуры ввода-вывода.

3. \*\*Вытесняющая многозадачность:\*\*

- Операционная система передает управление от одной программы к другой по различным событиям.

- Преимущества: Быстрый отклик на действия пользователя, полная реализация многозадачного ввода-вывода в ядре ОС. Недостатки: Требует дисциплины при написании кода, особые требования к реентерабельности и защите данных.

Реентерабельность программы означает, что её код может быть безопасно использован несколькими пользователями или процессами. Для обеспечения реентерабельности необходимо, чтобы вызываемый код не модифицировался, процедура не сохраняла информацию между вызовами, данные были уникальными для каждого пользователя, и не возвращались указатели на общие объекты. Реентерабельность важна для безопасности функций в многопоточных средах и программирования многозадачных систем, включая операционные системы.

**12. Иерархия, приоритеты и планирование потоков. Динамические уровни приоритетов.**

1. \*\*Иерархия и приоритеты потоков:\*\*

- \*\*Иерархия:\*\* Потоки часто группируются по уровням приоритета в зависимости от их важности и роли. Например, системные потоки могут иметь более высокий приоритет по сравнению с прикладными.

- \*\*Приоритеты:\*\* Каждому потоку присваивается приоритет, который определяет его относительную важность и влияет на решение о том, какие потоки будут выполнены в первую очередь.

2. \*\*Планирование потоков:\*\*

- \*\*Динамические уровни приоритетов:\*\* Некоторые системы поддерживают динамическое изменение приоритетов в зависимости от активности и состояния потоков. Например, при потреблении процессорного времени поток может повысить свой приоритет для более частого выполнения.

Цель планирования потоков вполне очевидна — определение порядка выполнения потоков в условиях внешней или внутренней многозадачности. Однако способы достижения этой цели существенно зависят от типа ОС. Рассмотрим сначала принципы планирования для универсальных ОС. Для таких ОС нельзя заранее предсказать, сколько и какие потоки будут запущены в каждый момент времени и в каких состояниях они будут находиться. Поэтому планирование должно выполняться динамически на основе сбора и анализа информации о текущем состоянии вычислительной системы.

\*\*Динамические уровни приоритетов в Windows:\*\*

В Windows каждый поток обладает динамическим приоритетом, который система может автоматически изменять для обеспечения отзывчивости и предотвращения голодания потоков. Динамический приоритет изначально совпадает с базовым приоритетом, и его система может повышать или понижать в зависимости от различных условий. Повышение приоритета происходит, например, при перемещении процесса на передний план, получении входных данных или выполнении условий ожидания. После повышения динамического приоритета система постепенно его уменьшает с течением времени. Этот механизм направлен на предотвращение инверсии приоритета и гарантирование эффективного использования процессорного времени.

**13. Синхронизация и взаимоблокировка ресурсов. Механизмы синхронизации.**

В некоторых операционных системах процессы, работающие совместно, могут сообща использовать некое общее хранилище данных. Каждый из процессов может считывать из общего хранилища данных и записывать туда информацию. Это хранилище представляет собой участок в основной памяти (возможно, в структуре данных ядра) или файл общего доступа. Местоположение совместно используемой памяти не влияет на суть взаимодействия и возникающие проблемы. Ситуации, в которых два (и более) процесса считывают или записывают данные одновременно и конечный результат зависит от того, какой из них был первым, называются состояниями состязания.

Критическая область — часть программы, в которой есть обращение к совместно используемым данным. Соответственно, критический ресурс — тот ресурс, к которому осуществляется одновременный доступ

\*\*Механизмы синхронизации:\*\*

1. \*\*Критические секции:\*\*

- Области кода, где доступ к общим данным ограничивается, чтобы предотвратить конфликты.

- Предоставляют мьютекс для защиты от одновременного доступа нескольких потоков.

2. \*\*Мьютексы:\*\*

- Обеспечивают механизм блокировки, позволяющий потоку получить доступ к ресурсу или критической секции.

- Предотвращают конфликты и обеспечивают монопольный доступ к ресурсу.

3. \*\*Семафоры:\*\*

- Имеют счетчик, контролирующий количество потоков, которым разрешен доступ к ресурсу.

- Подходят для сценариев с ограниченным числом доступных ресурсов.

4. \*\*События:\*\*

- Используются для сигнализации между потоками.

- Могут быть использованы для синхронизации, где один поток ждет наступления события, а другой его сигнализирует.

5. \*\*Барьеры:\*\*

- Используются для синхронизации группы потоков, ожидающих друг друга перед выполнением следующего этапа работы.

6. \*\*Атомарные операции:\*\*

- Гарантируют неделимость выполнения операции, предотвращая переключение контекста между потоками.

- Обеспечивают безопасное выполнение операций над общими данными без необходимости блокировки.

**14. Взаимоблокировка ресурсов в многозадачных системах. Решение задачи взаимоблокировки ресурсов.**

Взаимоблокировка в группе процессов возникает в том случае, если каждый процесс из этой группы ожидает события, наступление которого зависит исключительно от другого процесса из этой же группы.

для возникновения ресурсных взаимоблокировок должны выполняться четыре условия:

1. Условие взаимного исключения. Каждый ресурс либо выделен в данный момент только одному процессу, либо доступен.

2. Условие удержания и ожидания. Процессы, удерживающие в данный момент ранее выделенные им ресурсы, могут запрашивать новые ресурсы.

3. Условие невыгружаемости. Ранее выделенные ресурсы не могут быть принудительно отобраны у процесса. Они должны быть явным образом высвобождены тем процессом, который их удерживает.

4. Условие циклического ожидания. Должна существовать кольцевая последовательность из двух и более процессов, каждый из которых ожидает высвобождения ресурса, удерживаемого следующим членом последовательности.

*Решение задачи взаимоблокировки ресурсов*

Чаще всего для борьбы с взаимными блокировками используются четыре стратегии: 1. Игнорирование проблемы. Может быть, если вы проигнорируете ее, она проигнорирует вас.

2. Обнаружение и восстановление. Дайте взаимоблокировкам проявить себя, обнаружьте их и выполните необходимые действия.

3. Динамическое уклонение от них за счет тщательного распределения ресурсов.

4. Предотвращение за счет структурного подавления одного из четырех условий, необходимых для их возникновения.

**15. Компьютерное время. Ожидаемые таймеры.**

Время Windows — это количество времени в миллисекундах, прошедшее с момента последнего запуска системы. Этот формат в первую очередь предназначен для обеспечения обратной совместимости с 16-разрядной версией Windows. Чтобы обеспечить успешное выполнение приложений, предназначенных для 16-разрядной версии Windows, функция **GetTickCount()** возвращает текущее время Windows.

Хотя система внутренне использует время в формате UTC, в приложениях обычно отображается местное время, которое является датой и временем суток для вашего часового пояса. Поэтому, чтобы обеспечить правильные результаты, необходимо знать, ожидает ли функция получать время в формате UTC или местное время, а также возвращает ли функция время в формате UTC или местное время.

Объект таймера ожидания — это объект синхронизации, состояние которого по достижении указанного срока устанавливается в значение Signaled. Существует два типа таймеров ожидания, которые можно создать: сброс вручную и синхронизация. Таймер любого типа также может быть периодическим. Таймер сброса вручную – таймер, состояние которого остается сигнальным до вызова **SetWaitableTimer**(), чтобы установить новое время выполнения. Таймер синхронизации – таймер, состояние которого остается сигнальным до тех пор, пока поток не завершит операцию ожидания в объекте таймера. Периодический таймер – таймер, который повторно активируется каждый раз, когда истечет указанный период, пока таймер не будет сброшен или отменен. Периодический таймер — это либо периодический таймер сброса вручную, либо периодический таймер синхронизации. Поток использует функцию **CreateWaitableTimer** или **CreateWaitableTimerEx** для создания объекта таймера. Поток создания указывает, является ли таймер таймером сброса вручную или таймером синхронизации. Создающий поток может указать имя объекта таймера. Потоки в других процессах могут открывать дескриптор для существующего таймера, указывая его имя в вызове функции **OpenWaitableTimer**. Любой поток с дескриптором объекта таймера может использовать одну из функций ожидания для ожидания, пока состояние таймера будет задано как сигнальное. Поток вызывает функцию **SetWaitableTimer** для активации таймера.

\*\*Компьютерное время в программировании:\*\*

1. \*\*Системное Время:\*\*

- \*\*Определение времени:\*\* Компьютеры поддерживают внутренние часы, отслеживающие текущее системное время.

- \*\*RTC (Real-Time Clock):\*\* Часы в реальном времени, хранящие время даже при выключенном компьютере.

2. \*\*Работа с Временем в Программах:\*\*

- \*\*API и Библиотеки:\*\* Языки программирования предоставляют API и библиотеки для работы с временем (например, `time` в Python, `java.time` в Java).

- \*\*Типы данных времени:\*\* Обычно используются структуры данных, представляющие момент времени, интервалы и т. д.

3. \*\*Форматы Времени:\*\*

- \*\*UNIX Timestamp:\*\* Количество секунд, прошедших с 1 января 1970 года (эпоха) в UNIX-подобных системах.

- \*\*Дата и Время в Календарной Нотации:\*\* Представление даты и времени в удобочитаемой форме (например, "2023-01-01 12:00:00").

4. \*\*Операции с Временем:\*\*

- \*\*Арифметика Времени:\*\* Добавление или вычитание интервалов времени.

- \*\*Сравнение Времени:\*\* Операции сравнения для определения порядка событий.

- \*\*Форматирование и Разбор Времени:\*\* Преобразование времени в строковый формат и обратно.

5. \*\*Часовые Пояса и Летнее/Зимнее Время:\*\*

- \*\*TimeZone:\*\* Учет различий в часовых поясах.

- \*\*DST (Daylight Saving Time):\*\* Учет переходов на летнее/зимнее время.

6. \*\*Таймеры и Задержки:\*\*

- \*\*Таймеры ОС:\*\* Использование системных таймеров для планирования событий.

- \*\*Задержки:\*\* Ожидание выполнения определенного условия или прошествия времени.

7. \*\*Системные Часы и Синхронизация:\*\*

- \*\*NTP (Network Time Protocol):\*\* Протокол синхронизации времени через сеть.

- \*\*Системные События:\*\* Обновление системного времени при включении, сетевых событиях и др.

**Тема 3.1. Управление памятью**

**16. Управление памятью: адресное пространство процесса, организация памяти, основные механизмы управления памятью, концепция рабочего множества.**

Адресное пространство  
Каждое приложение (или процесс) в операционной системе имеет свое собственное адресное пространство. Это означает, что каждому процессу назначается уникальный набор адресов памяти, которые он может использовать.   
Это помогает изолировать процессы друг от друга и предотвращает случайное или намеренное повреждение данных одного процесса другим процессом.  
адреса, которые видит приложение, являются адресами виртуальной памяти.  Операционная система и аппаратное обеспечение компьютера переводят эти виртуальные адреса в физические адреса памяти.

**Адресное пространство** - это набор адресов, который может быть использован процессом для обращения к памяти.   
У каждого процесса имеется собственное адресное пространство, независимое от того адресного пространства, которое принадлежит другим процессам (за исключением тех особых обстоятельств, при которых процессам требуется совместное)  
Также оно может быть частным,а может быть общим (если оно не является общим(обычно относится к ситуации, когда два или более процесса имеют доступ к одним и тем же адресам в памяти. Это означает, что эти процессы могут читать и записывать в одни и те же места в памяти. Это может быть полезно для обмена данными между процессами)  
)  
  
**организация памяти**

Организация памяти в компьютере - это способ, которым операционная система управляет и контролирует физическую и виртуальную память.

Основные функции управления памятью включают:

-Отслеживание свободной и занятой памяти.

-Выделение памяти процессам при их создании и освобождение памяти при их завершении.

-Вытеснение процессов из оперативной памяти на диск, когда размеры основной памяти недостаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место.

-Настройка адресов программы на конкретную область физической памяти.

Физическая память компьютера имеет иерархическую структуру и

Виртуальная память позволяет программам использовать больше памяти, чем физически доступно

**основные механизмы управления памятью**

Основные механизмы управления памятью в операционных системах включают:

-Отслеживание свободной и занятой памяти: Операционная система должна знать, какая память в данный момент используется, и какая свободна.

-Выделение памяти процессам: Когда процесс создается, ему нужно выделить определенное количество памяти для его работы.

-Освобождение памяти при завершении процессов: Когда процесс завершается, выделенная ему память должна быть освобождена и возвращена в общий пул.

-Вытеснение процессов из оперативной памяти на диск: Если в оперативной памяти не хватает места для всех процессов, некоторые из них могут быть временно выгружены на диск.

-Возвращение процессов в оперативную память: Когда в оперативной памяти освобождается место, процессы, выгруженные на диск, могут быть возвращены обратно.

-Настройка адресов программы на конкретную область физической памяти: Каждый процесс в системе должен иметь свое собственное адресное пространство, которое отображается на физическую память.

Эти механизмы обеспечивают эффективное использование доступной памяти и поддерживают многозадачность и защиту памяти  
-----  
Представьте, что у вас есть процесс, который выполняет некоторую задачу в операционной системе. Этот процесс может обращаться к различным страницам памяти в течение своего выполнения.

**Рабочее множество этого** процесса - это понятие, которое определяет количество памяти, требующееся процессу в заданный интервал времени. Другими словами, рабочее множество процесса - это набор страниц в виртуальном адресном пространстве процесса, которые в данный момент находятся в физической памяти

Важно отметить, что рабочее множество является динамическим и может меняться со временем в зависимости от поведения процесса.  
*Обычно за единицу информации берется страница памяти*   
( **Представьте**, что у вас есть программа, которая обрабатывает большой массив данных. В течение определенного интервала времени эта программа может работать только с небольшим подмножеством этих данных. Это подмножество данных, которые программа использует в данный момент, и составляет рабочее множество процесса)  
  
**Отводка:** *Страница в памяти* - это блок фиксированного размера, на которые делится как физическая, так и виртуальная память. Это ключевой элемент технологии управления памятью, известной как страничная память(**пример**: Допустим, у вас есть программа, которая работает с большим массивом данных. Весь этот массив не может быть загружен в оперативную память одновременно из-за ограниченности ресурсов. Здесь вступает в игру система управления памятью на основе страниц.

Вместо того чтобы загружать весь массив данных, операционная система загружает только те “страницы” данных, которые в данный момент нужны программе. Каждая страница - это блок данных фиксированного размера.

Когда программа обращается к данным, которые в данный момент не находятся в оперативной памяти (это называется промахом страницы), операционная система выбирает страницу для выгрузки (если все страницы в памяти заняты), сохраняет ее на диск и загружает нужную страницу в оперативную память12.

Этот процесс называется подкачкой страниц и позволяет эффективно использовать ограниченные ресурсы памяти, давая программам возможность работать с объемами данных, превышающими размер доступной оперативной памяти)

**17. Классификация запоминающих устройств. Иерархия памяти. Оперативные и постоянные запоминающие устройства.**

-Классификация запоминающих устройств.

Запоминающее устройство — носитель информации, предназначенный для записи и хранения данных. В основе работы запоминающего устройства может лежать любой физический эффект, обеспечивающий приведениесистемы к двум или более устойчивым состояниям.  
Разделяется по устойчивости записи и возможности перезаписи:

**-Постоянные** (ПЗУ), содержание которых не может быть изменено конечным пользователем (например, BIOS). ПЗУ в рабочем режиме допускает только считывание информации.  
(Это как книга, написанная заводом-изготовителем. Вы можете читать информацию из нее, но не можете внести изменени)

-**Записываемые** (ППЗУ), в которые конечный пользователь может записать информацию только один раз (например, CD-R)( Это как бланк, на котором вы можете написать информацию только один раз. После записи информации она не может быть изменена)

-**Многократно перезаписываемые** (ПППЗУ) (например, CD-RW).( Это как доска для рисования, на которой вы можете рисовать, стирать и снова рисовать многократно)

-**Оперативные** (ОЗУ) — обеспечивают режим записи, хранения и считывания информации в процессе её обработки. Быстрые, но дорогие ОЗУ (SRAM) строят на триггерах, более медленные, но более дешёвые разновидности ОЗУ — динамические ЗУ (DRAM) строят на элементах, состоящих из ёмкости (конденсатора) и полевого транзистора, используемого в качестве ключа разрешения записи-чтения. В обоих видах ЗУ информация исчезает после отключения от источника питания (например, тока).( Это как ваш рабочий стол, на котором вы работаете с информацией. Вы можете положить что-то на стол, работать с этим, а затем убрать)

**По типу доступа ЗУ делятся на:**

-**устройства с последовательным доступом** (например, магнитные ленты).( Это как кассета, где вы должны пройти через все песни, чтобы добраться до нужной.)

-**устройства с произвольным доступом** (RAM) (например, оперативная память)( : Это как книга, где вы можете открыть любую страницу напрямую2).

-**устройства с прямым доступом** (например, жесткие магнитные диски).( Это как DVD, где вы можете быстро перейти к определенной главе.)

-**устройства с ассоциативным доступом** (специальные устройства, для повышения производительности БД(Это специальное устройство, которое позволяет быстро найти информацию по определенному критерию)(ассоциативная память (пользователь задавал слово данных и ассоциативная память осуществляла его поиск чтобы выяснить хранится ли оно где-либо в памяти.Если слово данных найдено ассоциативная память возвращает список 1 или более адресов где слово было найдено))

Отличие **устройства с произвольным доступом от устройства с прямым доступом** (Устройства с произвольным доступом, такие как оперативная память (RAM), обеспечивают значительно более быстрый доступ к данным по сравнению с устройствами с прямым доступом, такими как жесткие диски. Это связано с тем, что RAM использует полупроводниковые технологии, которые позволяют быстро получать и записывать данные, в то время как жесткие диски используют механические компоненты, которые работают значительно медленнее. Однако жесткие диски обычно имеют гораздо большую емкость и используются для долгосрочного хранения данных)

**Классификация** запоминающих устройств по геометрическому исполнению: (относится к физической форме или структуре запомниющего устройства)  
-дисковые(CD \DVD для хранения музыки или фильмов ) (магнитные диски, оптические, магнитооптические);

-ленточные(аудиокасеты) (магнитные ленты, перфоленты);   
-барабанные (это старый тип запоминающего устройства, который использовал вращающийся барабан с магнитным покрытием)(магнитные барабаны);

-карточные(кредитные карты млм флэш-карты) (магнитные карты, перфокарты, флэш-карты, и др.)   
-печатные платы(как оперативная память RAM в компбютере) (карты DRAM).

**Классификация** запоминающих устройств по физическому принципу(относится к методу или технологии, используемой для хранения и извлечения информации.):   
-перфорационные перфокарта; перфолента (используют отверстия в материале для кодирования информации.);

-с магнитной записью (ферритовые сердечники, магнитные диски, магнитные ленты, магнитные карты);( используют магнитные поля для хранения информации.)

-оптические (CD, DVD, HD-DVD, Blu-ray Disc);( используют лазеры для чтения и записи информации.)

-использующие эффекты в полупроводниках (флэш-память) и другие.( такие как флэш-память, используют электронные компоненты для хранения информации.)USB-накопители\SSD\карты памяти

**Иерархия памяти.**

Современная система памяти образует иерархию от быстрых типов памяти маленького размера до медленных типов памяти большого размера. Мы говорим, что конкретный уровень иерархии кэширует или является кэшем для данных, расположенных на более низком уровне. Это значит, что он содержит копии данных с более низкого уровня. Когда

процессор хочет получить какие-то данные, он их сперва ищет на самых быстрых высоких уровнях. И спускается на более низкие, если не может найти  
 высокий уровень содержит все данные маленьких. Когда идет поиск сначала ищет на самых больших а потом идет к маленьким  
  
На вершине иерархии находятся регистры ( представляют собой сверхбыструю память внутри процессора)процессора. Доступ к ним занимает 0 тактов*(определённое количество тактовых циклов чтобы получить доступ к чему-то),* но их всего несколько штук.

Далее идёт несколько килобайт кэш-памяти первого уровня(L1), доступ к которой занимает примерно 4 такта.(Потом идёт пара сотен килобайт более медленной кэш-памяти второго уровня(L2).Потом несколько мегабайт кэш-памяти третьего уровня(L3)*.( Она гораздо медленней, но всё равно быстрее оперативной памяти*))  
Далее расположена относительно медленная оперативная память.

Оперативную память *можно рассматривать как кэш для локального диска* (локальные диски тоже можно разделить на отдельные категории)  
Локальный диск сам может рассматриваться как кэш для данных, расположенных на удалённых серверах. *Резервные копии данных можно хранить на магнитных лента.(*Это как хранилище для резервных копий. Это может занять много времени, *)***На примерах:**  
Регистры процессора: Это как ящики на вашем рабочем столе, в которых вы храните вещи, с которыми работаете прямо сейчас. Они очень быстрые, но их мало.

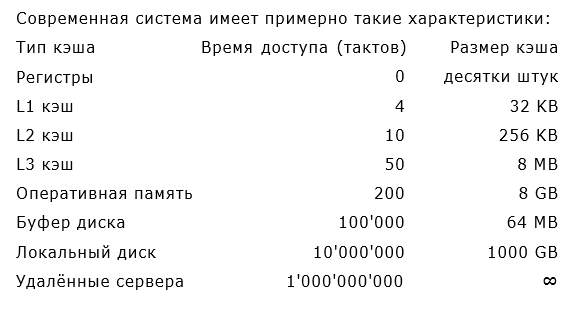
Кэш-память первого уровня (L1): Это как полка над вашим столом. Доступ к ней занимает немного больше времени, чем к ящикам, но она все равно очень быстрая.

Кэш-память второго уровня (L2) и третьего уровня (L3): Это как шкафы в вашем кабинете. Они медленнее, чем полка, но быстрее, чем выходить из кабинета.

Оперативная память (RAM): Это как архив в вашем офисе. Он медленнее, чем шкафы, но быстрее, чем ехать в удаленный склад.

Локальный диск: Это как удаленный склад. Он медленнее, чем архив, но быстрее, чем заказывать что-то из другого города.

Удаленные серверы: Это как заказывать что-то из другого города. Это может занять много времени, но вы можете получить почти все, что вам нужно.

Магнитные ленты: Это как хранилище для резервных копий. Это может занять много времени, но это хороший способ сохранить информацию на случай, если что-то пойдет не так.  
**Описание характеристик различных типов памяти в современной компьютерной системе**   


**Оперативные и постоянные запоминающие устройства**Оперативная память (также оперативное запоминающее устройство, ОЗУ) -

предназначена для временного хранения данных и команд, необходимых процессору для выполнения им операций. Оперативная память передаёт процессору данные непосредственно, либо через кэш-память. Каждая ячейка оперативной памяти имеет свой индивидуальный адрес  
  
Оперативное запоминающее устройство (ОЗУ) - это тип памяти, который используется компьютером для временного хранения данных и команд, необходимых процессору для выполнения операций  
(**Пример :**Это как рабочий стол, на котором вы работаете с документами. Вы можете быстро получить доступ к этим документам, но если вы выключите компьютер, все данные на рабочем столе исчезнут. ОЗУ используется для временного хранения данных, которые активно используются)

ПЗУ (Постоянное запоминающее устройство) - это тип памяти в компьютере, используется для хранения информации, которая не меняется и которая нужна компьютеру для работы, например, программное обеспечение для загрузки операционной системы  
( **Пример:**Это как шкаф для хранения файлов. Вы можете сохранять документы в шкафу на неопределенное время, даже если вы выключите компьютер.)  
**Что там хранится** :  
Программа управления работой процессора.

Программа запуска и останова компьютера.

Программы тестирования устройств.

Программы управления дисплеем, клавиатурой, принтером, внешней памятью.

Информация о том, где на диске находится операционная система.  
  
Тема 3.2. Организация виртуальной памяти

**18. Виртуальная память. Структуризация адресного пространства виртуальной памяти. Задачи управления виртуальной памятью.**  
Виртуальная память - это метод управления памятью компьютера, который позволяет выполнять программы, требующие больше оперативной памяти, чем имеется в компьютере( Это достигается за счет автоматического перемещения данных из оперативной памяти в файл подкачки, расположенный на жестком диске)  
  
Виртуальная память решает следующие задачи:   
-размещает данные в запоминающих устройствах разного типа, например, часть программы в оперативной памяти, а часть на диске;   
-перемещает по мере необходимости данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы с диска в оперативную память;  
- преобразует виртуальные адреса в физические.  
Все эти действия выполняются автоматически, без участия программиста, то есть механизм виртуальной памяти является прозрачным по отношению к пользователю.  
  
Процессоры архитектуры x86-64 поддерживают два основных режима работы: Long mode («длинный» режим) и Legacy mode («унаследованный», режим совместимости с 32-битным x86)1. Эти режимы влияют на структуризацию адресного пространства виртуальной памяти.

Long mode («длинный» режим): Это «родной» режим для процессоров x86-64, который позволяет использовать все преимущества 64-битной архитектуры. В этом режиме можно выполнять 64-битные программы, а также поддерживается выполнение 32-битного кода для обратной совместимости.

Legacy mode («унаследованный» режим): Этот режим позволяет процессору работать как 32-битный процессор x86, обеспечивая полную совместимость с 32-битным кодом и операционными системами. В этом режиме 64-битные программы и операционные системы работать не будут  
  
Виртуальная память позволяет компьютеру использовать больше памяти, чем физически доступно, путем перемещения данных между оперативной памятью и дисковым пространством. Структуризация адресного пространства виртуальной памяти включает разделение его на части и динамическую трансляцию адресов(позволяет выполнять программы размер которых превышает размер оперативной памяти)  
**На примере :**  
Long mode («длинный» режим): Это как если бы у вас был суперкар, который может развивать огромную скорость и имеет множество дополнительных функций. Но чтобы использовать все эти функции, вам нужна специальная дорога (64-битная операционная система). В этом режиме вы можете использовать все преимущества своего суперкара1.

Legacy mode («унаследованный» режим): Это как если бы вы водили свой суперкар по обычной дороге с ограничением скорости. Вы не можете использовать все его функции, но зато вы можете ездить по любым дорогам (совместимость с 32-битными приложениями и операционными системами)1.

Эти режимы влияют на то, как компьютер “видит” и использует память. В “длинном” режиме компьютер может “видеть” и использовать больше памяти, чем в “унаследованном” режиме

Также к структуризации можно отнести :  
Структуризация адресного пространства виртуальной памяти включает в себя различные методы, такие как сегментация и страничная адресация3.

Сегментация: Виртуальное адресное пространство разделено на сегменты, каждый из которых имеет свой уникальный адрес. Сегмент - это условно выделенная область адресного пространства определенного размера.

Страничная адресация: Виртуальная и физическая память разбиваются на блоки фиксированной длины, называемые страницами. Страницы виртуальной и физической памяти нумеруются. Преобразование виртуальных адресов в физические выполняется с помощью таблицы страниц

**19. Подкачка. Алгоритмы замещения страниц. Куча (heap). Стек.**Файл подкачки или виртуальная память — это способ системы виртуальной памяти увеличить оперативную память, когда ее не хватает для совершения операций. Система автоматически задействует файл подкачки, когда приложениям не хватит системной памяти ОЗУ. Хотя система сама регулирует объем файла подкачки иногда может понадобиться вручную увеличить виртуальную память  
  
Когда в ОЗУ заканчивается место, операционная система перемещает некоторые данные из ОЗУ в файл подкачки. Это освобождает место в ОЗУ для выполнения текущих задач.

Если эти данные снова становятся необходимыми, они возвращаются обратно в ОЗУ. Этот процесс называется "вытеснением  
*Может замедлить работу компьютера, поскольку чтение и запись данных на жесткий диск медленнее, чем в ОЗУ*  
(вWindows файл подкачки управляется автоматические ,но его можно настроить вручную )  
Подкачка в Linux - это процесс, при котором операционная система перемещает данные между оперативной памятью и диском, чтобы обеспечить эффективное использование памяти1. Вместо перемещения целых процессов, как это было в ранних системах UNIX, Linux работает с отдельными страницами памяти.

Основная идея подкачки в Linux заключается в том, что процессу не обязательно находиться целиком в памяти для того, чтобы выполняться. Вместо этого, страницы данных, необходимые для выполнения процесса, подкачиваются динамически по мере необходимости.

Демон страниц - это специальный процесс в Linux, который периодически проверяет, необходимо ли освободить некоторую память. Если доступной памяти становится недостаточно, демон страниц начинает освобождать страницы  
  
В Windows есть три основных файла подкачки: hiberfile.sys, pagefile.sys и swapfile.sys1hiberfile.sys - это файл для сохранения памяти в режиме «сон» (гибернация); pagefile.sys - основной файл подкачки; swapfile.sys - файл подкачки отдельных  
 отличия подкачки в linux от пдкачки windows\  
-Страницы памяти: **(одинаково)**  
В Linux, подкачка работает с отдельными страницами памяти, а не с целыми процессами. Это означает, что только те части процесса, которые действительно нужны, загружаются в память. В Windows, страницы также используются для подкачки, но подход к управлению этими страницами может отличаться1.

-Файл подкачки **(одинаково)**  
 В Linux, файл подкачки используется только тогда, когда это действительно необходимо. Если общий объем виртуальной памяти меньше, чем имеющаяся физическая память, то файл подкачки не нужен. В Windows, файл подкачки используется более активно и может быть распределен по нескольким дискам для повышения производительности.

-Управление подкачкой:

В Linux, управление подкачкой осуществляется ядром и специальным процессом, называемым демоном страниц. В Windows, управление подкачкой также осуществляется операционной системой, но подходы к управлению могут отличаться.  
  
**Алгоритмы замещения страниц** - это методы, которые операционная система использует для определения, какую страницу памяти следует удалить, когда требуется место для новой страницы.   
Вот некоторые из наиболее важных алгоритмов:

-Оптимальный алгоритм замещения страниц: Выбирает для удаления страницу, которая будет использоваться позже всего.(которая не будет использоваться в ближайшее время)

-Алгоритм исключения недавно использовавшейся страницы (LRU): Выбирает для удаления страницу, которая не использовалась дольше других.

-Алгоритм FIFO (First In, First Out): Выбирает для удаления страницу, которая находилась в памяти дольше других.

-Алгоритм «второй шанс»: Модификация алгоритма FIFO, которая проверяет, была ли страница недавно использована перед ее удалением.

-Алгоритм «часы»: Еще одна модификация алгоритма FIFO, которая использует “стрелку часов” для отслеживания страниц.(***Выглядит в виде круга вместо цифр -страницы*** Если у этой страницы установлен бит обращения (то есть она недавно использовалась), алгоритм сбрасывает бит обращения и перемещает “стрелку” к следующей странице. Если бит обращения не установлен (то есть страница не использовалась недавно), алгоритм заменяет эту страницу.)

-Алгоритм замещения наименее востребованной страницы (LFU): Выбирает для удаления страницу, которая использовалась реже всего.

Алгоритм LRU обычно работает хорошо, но может быть дорогим в реализации. Алгоритм FIFO прост в реализации, но может привести к частым ошибкам отсутствия страниц  
  
Куча(выбираем что удалить) хранятся глобальные переменные   
“Куча” в программировании - это область памяти, которую можно динамически управлять.   
Это как свободный пул памяти, который вы можете использовать, когда запускаете приложение. Размер кучи определяется ограничениями вашей оперативной памяти и обычно больше, чем размер стека.  
Мы используем кучу, когда не знаем, сколько памяти займет структура данных в нашей программе, когда нам нужно выделить больше памяти, чем доступно в стеке, или когда нам нужно создать переменные, которые будут существовать в течение всего времени работы приложения

В Windows каждый процесс имеет кучу по умолчанию, предоставляемую системой. Приложения, которые часто используют кучу, могут повысить производительность с помощью частных куч(это блок памяти в адресном пространстве процесса.).  
функции HeapAlloc и HeapFree для управления памятью(является непрерывной) в этой куче.  
Функция HeapCreate создает частный объект кучи, из которого процесс может выделять блоки памяти(указывает начальный и максимальный размер кучи.)  
Функция HeapDestroy уничтожает частный объект кучи. Он удаляет и освобождает все страницы объекта кучи, а также делает дескриптор кучи недействительным.  
Память из кучи выделяется с помощью функции malloc(). malloc возвращает указатель void, который затем нужно привести к правильному типу. После использования блока памяти его нужно освободить помощью функции free().  
 **-Стек(**очищается автоматически ) хранятся локальные переменные Стек - это область памяти, где данные добавляются и удаляются по принципу “последним пришел - первым вышел” (LIFO). Это значит, что вы можете получить доступ только к последнему элементу, который был добавлен в стек.  
Если в стек записывается слишком много данных, он может “вырасти” за свои пределы, что приводит к переполнению стека. Это может вызвать “синий экран смерти” в Windows. **Современные операционные системы** используют улучшенную проверку границ и восстановление ошибок, чтобы уменьшить вероятность переполнения стека.  
Чтение из стека называется «извлечение» (pop), а запись в стек — «вталкивание» (push).  
**Все архитектуры x86** используют стек как временное хранилище в оперативной памяти, что позволяет процессору быстро сохранять и извлекать данные. Стек “растет” вниз, от верхних адресов памяти к младшим. *В архитектуре x86-64 количество регистров общего назначения увеличилось вдвое по сравнению с x86***20. Тема 3.3. Управление вводом-выводом и файлами  
Типы устройств ввода/вывода. Обработка внешних прерываний. Синхронный и асинхронный ввод/вывод.**

Устройства ввода/вывода — компонент типовой архитектуры ЭВМ, предоставляющий компьютеру возможность взаимодействия с внешним миром и, в частности, с пользователями.   
По направлению передачи информации подразделяются на:   
-устройства ввода, (клавиатура\мышь\тачпад)  
-устройства вывода, (монитор \принтер\проектор)  
-устройства ввода и вывода(дисковод\модем) обычно состоят из механического и электронного компонента, где электронный компонент называется контроллером устройства.  
**Разделяются** условно  
блочные устройства и символьные устройства.  
-К блочным относятся такие устройства, которые хранят информацию в блоках фиксированной длины, у каждого из которых есть собственный адрес.  
. Важным свойством блочного устройства является то, что оно способно читать или записывать каждый блок независимо от всех других блоков. Среди наиболее распространенных блочных устройств жесткие диски, приводы оптических дисков и флеш-накопители USB.  
-Символьные устройства. Они выдают или воспринимают поток символов, не относящийся ни к какой блочной структуре. Они не являются адресуемыми и не имеют никакой операции позиционирования. В качестве символьных устройств могут рассматриваться терминалы, принтеры(**ПРИМЕР** Когда вы отправляете документ на печать, он сначала преобразуется в формат, который принтер может понять. Этот формат, обычно называемый языком описания страницы (Page Description Language, PDL), затем передается принтеру по байтам. Принтер интерпретирует эти байты и преобразует их в физические движения печатающей головки, чтобы создать изображение на бумаге),  
 сетевые интерфейсы, мыши.( это тип устройств, с которыми можно взаимодействовать как с потоком байтов. Это означает, что данные передаются по одному байту за раз, что делает символьные устройства идеальными для устройств, которые обрабатывают данные этим образом, например, клавиатуры)  
  
Обработка внешних прерываний  
Внешние прерывания - это сигналы, которые поступают к процессору от внешних устройств, таких как периферийные устройства.  
ти сигналы могут произойти в любой произвольный момент и обычно связаны с событиями, такими как сигнал от таймера, сетевой карты или дискового накопителя, нажатие клавиш клавиатуры, движение мыши  
  
Обслуживание прерывания состоит из двух, а иногда и трех этапов.  
-Быстрое сохранение непостоянной информации: Когда происходит прерывание, процессор быстро сохраняет непостоянную информацию, такую как содержимое регистра, в процедуре прерывания. Это происходит на уровне запроса прерывания (IRQL) DIRQL.( , процессор быстро сохраняет важную информацию, которую он в данный момент обрабатывает, чтобы вернуться к ней позже.)

-Обработка сохраненных переменных данных: Затем процессор обрабатывает сохраненные переменные данных в отложенном вызове процедуры (DPC), который выполняется на уровне запроса прерывания DISPATCH\_LEVEL(затем процессор начинает обрабатывать прерывание. Это может включать в себя чтение данных, которые были отправлены устройством, или выполнение каких-то действий в ответ на прерывание.).

-Выполнение дополнительных работ: Если это необходимо, процессор выполняет дополнительные работы на уровне запроса прерывания PASSIVE\_LEVEL.( если после обработки прерывания требуется выполнить еще какую-то работу, процессор может сделать это на третьем этапе.)  
 **Синхронный и асинхронный ввод/вывод**

Асинхронный ввод-вывод используется там, где можно оптимизировать производительность приложения.  
При асинхронном вводе-выводе приложение инициирует операцию ввода-вывода, а затем может продолжить свою работу (во время выполнения этого запроса).   
При синхронном вводе-выводе приложение блокируется до завершения выполнения операции ввода-вывода.   
С точки зрения вызывающего потока асинхронный ввод-вывод более эффективен, поскольку позволяет продолжать выполнение, в то время как операция ввода-вывода ставится диспетчером ввода-вывода в очередь и впоследствии выполняется.  
*Однако приложение, использующее асинхронный ввод-вывод, требует механизма определения завершенности этой операции.*В Unix-системах все файловые дескрипторы (это способы, которыми ваша программа взаимодействует с файлами и устройствами) по умолчанию работают в синхронном режиме.

Есть два способа сделать ввод-вывод более эффективным:

Неблокирующий режим ввода-вывода: Это когда файловый дескриптор настроен так, чтобы системные вызовы для ввода-вывода возвращали ошибку, если данные не готовы в момент вызова, вместо того чтобы “зависать” и ждать.

Мультиплексирование с помощью системного API: Это когда используется специальный системный вызов, который позволяет отслеживать приход данных для множества файловых дескрипторов одновременно. Это позволяет вашему приложению эффективно обрабатывать ввод-вывод с множества источников без необходимости “зависать” и ждать каждой операции.  
  
**21. Файловые системы**   
**Файловые системы. Файлы и директории. Управление внешней памятью.**

Файловая система (file system) — порядок, определяющий способ организации, хранения и именования данных на носителях информации в компьютерах, а также в другом электронном оборудовании: цифровых фотоаппаратах, мобильных телефонах и т. п.   
Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла.  
  
Вот некоторые из наиболее распространенных типов файловых систем:

NTFS: New Technology File System (NTFS) - это файловая система, разработанная Microsoft для операционной системы Windows. Она поддерживает метаданные и использует журналирование для повышения надежности.

FAT32: File Allocation Table (FAT) - это старая файловая система, которая все еще широко используется для совместимости, особенно на съемных накопителях.

exFAT: Extended File Allocation Table (exFAT) - это модернизированная версия FAT32, которая поддерживает большие файлы и большие тома.

EXT4: Fourth Extended Filesystem (EXT4) - это файловая система, широко используемая в Linux.

HFS+: Hierarchical File System Plus (HFS+) - это файловая система, разработанная Apple для использования в macOS.

APFS: Apple File System (APFS) - это файловая система, разработанная Apple для iOS и macOS, которая заменила HFS+.

Btrfs: B-tree File System (Btrfs) - это файловая система с открытым исходным кодом для Linux, которая поддерживает большое количество функций, включая снимки и сжатие2.

ZFS: Z File System (ZFS) - это файловая система, разработанная Sun Microsystems для операционной системы Solaris. Она поддерживает большое количество функций, включая снимки, копирование при записи и автоматическое реплицирование  
  
**Файлы и директории**   
Файл - поименованная совокупность данных, обычно размещаемая на внешних запоминающих устройствах.  
  
Директории, также известные как папки, используются для организации файлов. Они могут содержать файлы и другие директории, создавая иерархическую структуру, известную как файловая система. Директории помогают упорядочивать файлы, делая их поиск и управление более удобными  
(в Linux можно посмотреть содержимое директории используя команду ls\mkdirсоздать \ . В Windows вы можете просматривать и управлять файлами и директориями через Проводник Windows.)  
  
Расширения файлов: Многие программы ожидают, что имена файлов будут состоять из основного имени и расширения, разделенных точкой. Расширения имеют произвольную длину, и файлы могут иметь несколько расширений.

Каталоги: Каталоги используются для группировки файлов и могут содержать подкаталоги, создавая иерархическую файловую систему. Корневой каталог обозначается как “/”, и символ “/” также используется для разделения имен каталогов.

Абсолютные и относительные пути: В Linux можно использовать абсолютные пути, которые указывают, как найти файл от корневого каталога,   
или относительные пути, которые указываются относительно текущего рабочего каталога.(прописываем папка относитьльно какой-то программы )/source/img(unity)

***Ссылка*** - это способ указания на другой файл или директорию в файловой системе. Она представляет собой запись, которая идентифицирует другой файл или директорию. Ссылки позволяют размещать один и тот же файл в нескольких директориях.( это записи каталога, указывающие на существующие файлы. Они могут быть жесткими или символьными.) (Символическая ссылка - это отдельный файл, который указывает на другой файл или директорию. Жесткая ссылка - это прямая ссылка на физическое местоположение файла на диске.)

Скрытые файлы и домашние каталоги: Файлы, имена которых начинаются с точки, считаются скрытыми и не отображаются при обычном выводе команды ls. Домашний каталог пользователя предназначен для хранения личных данных пользователя и настроек для программ. Полный путь к домашнему каталогу хранится в переменной окружения HOME.  
  
**Точка .в начале имени файла делает его скрытым, то есть, он не показывается в выводе команды ls.  
  
Управление внешней памятью**Управление внешней памятью - это процесс, который организует эффективную работу с данными, хранящимися во внешней памяти, и обеспечивает пользователю удобный интерфейс при работе с такими данными.

В операционной системе Windows, управление внешней памятью осуществляется путем создания и управления файлом подкачки, также известного как файл подкачки (swap file) или страницы файла подкачки (page file)2.

В ранних операционных системах, управление памятью сводилось просто к загрузке программы и ее данных из некоторого внешнего накопителя (перфоленты, магнитной ленты или магнитного диска) в оперативную память3.

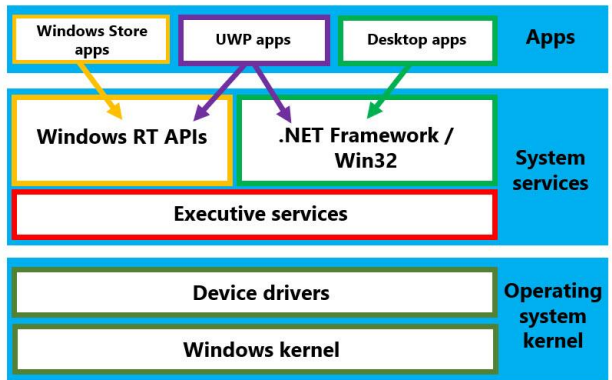
В Windows 10, вы можете управлять памятью через “Параметры” -> “Система” -> "Память устройства". Также можно настроить оперативную память через "Редактор реестра".

Важно отметить, что управление внешней памятью - это сложный процесс, который включает в себя множество аспектов, таких как управление памятью, управлен

**22. Принципы организации и структура ОС Windows. Обзор версий Windows. Методы инсталляции ОС Windows.**

**Принципы организации и структура ОС Windows**

Архитектура операционной системы Windows включает ядро операционной системы, системные службы и приложения



*Ядро операционной системы*

На самом низком уровне операционной системы ядро операционной системы состоит из самого ядра Windows и драйверов устройств низкого уровня. Ядро отвечает за прием запросов операционной системы от системных служб. Затем ядро преобразует эти запросы в инструкции для аппаратного обеспечения компьютера, включая центральный процессор (ЦП), память и аппаратные устройства. При запуске операционной системы сначала инициализируются ядро и связанные с ним низкоуровневые драйверы устройств, а затем службы операционной системы.

*Системные службы*

Службы операционной системы являются частью операционной системы, а не компонентами, которые вы устанавливаете после развертывания операционной системы. Кроме того, службы операционной системы функционируют без каких-либо действий со стороны пользователя. Фактически они начинаются до того, как пользователь войдет в систему. И службы операционной системы, и драйверы устройств являются программным обеспечением. Однако разница между ними заключается в том, что драйверы устройств напрямую взаимодействуют с аппаратными устройствами или компонентами. Как правило, системная служба взаимодействует с другими программными компонентами операционной системы

*Понимание приложений*

На верхнем уровне операционной системы приложения работают путем взаимодействия с пользователем компьютера, а на нижнем уровне — путем интеграции со службами операционной системы

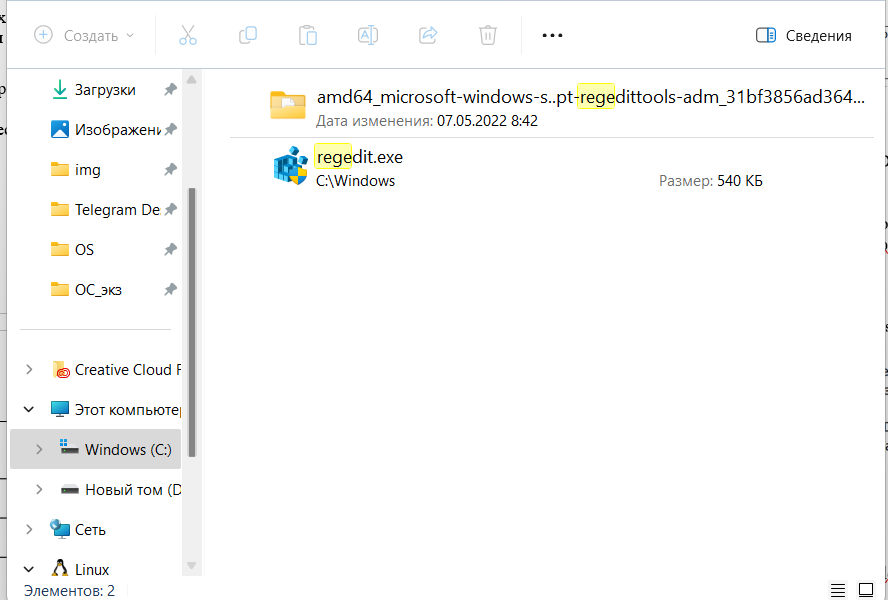
*Отличия службы от приложения*

Служба – процесс, работающий, даже когда никто не зарегистрирован в системе.

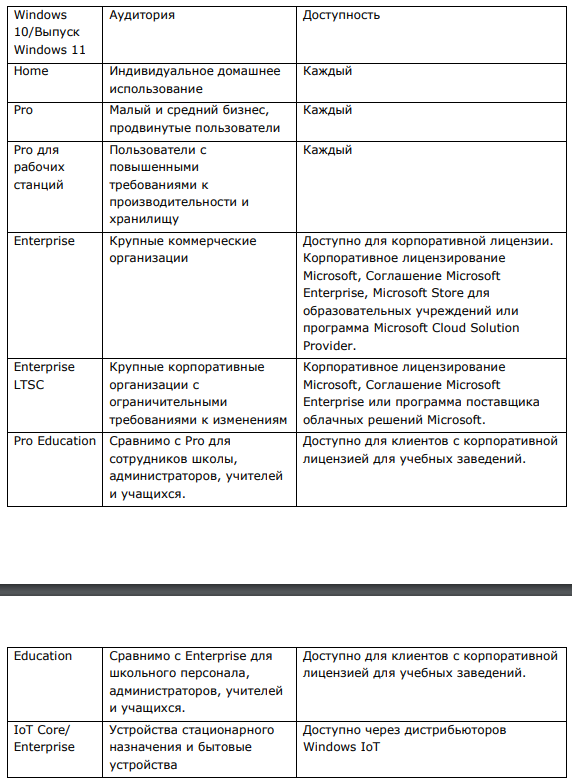
*Сравнение служб и драйверов*

Информация о службах и драйверах хранится в одном и том же разделе реестра. Различить их можно по параметрам Start и Type

*Реестр Windows*

Реестр — это база данных, в которой Windows хранит параметры конфигурации(набор параметров и настроек) пользователя и компьютера. Всякий раз, когда вы вносите изменения в конфигурацию Windows, это изменение фиксируется в реестре. Реестр Windows организован иерархически  


**Обзор версий Windows**



*Серверные версии Windows:*

Последняя LTSC-версия: Windows Server 2022, также поддерживается Windows Server 2019.

*Безопасность:*

Windows Server 2022 обеспечивает многоуровневую безопасность и гибридные возможности с Azure.

Встроенные возможности защищенного ядра снижают риск от угроз безопасности.

*Защищенное Соединение:*

Улучшенное защищенное соединение с использованием быстрых и защищенных зашифрованных соединений HTTPS и стандартного шифрования SMB AES 256.

*Редакции Windows Server 2022:*

Standard: Основная функциональность, исключая Hotpatching и SDN. Storage Replica ограничен до 2TB и 2 автоматически лицензируемых машин.

Datacenter: Поддерживает все функции, кроме Hotpatching и SMB over QUIC.

Datacenter: Azure Edition: Премиальная редакция, поддерживается в Azure, предлагает дополнительные функции, такие как горячее обновление и SMB через QUIC.

**Методы инсталляции ОС Windows(процесс установки ОС на компьютер)**

*Ручная установка с использованием установочного носителя(на флешке есть ос ):*

Установка системы вручную с использованием установочного носителя.

Применяется, когда нет подходящего образа для конкретной аппаратной платформы или при небольшом количестве установок.

*Автономная (необслуживаемая) установка(когда все необходимые файлы для установки уже находятся на установочном носителе):*

Используется файл ответов, предоставляющий ответы на вопросы, задаваемые во время ручной установки.

Позволяет автоматизировать установку с заданными параметрами.

*Установка, сопровождаемая производителем:*

Некоторые производители поставляют носители автономной установки, упрощающие процесс для конечного пользователя.

*Создание копий или образов систем:*

Полезно при развертывании нескольких идентичных компьютеров или серверов.

Создание образов системы с помощью инструментов Microsoft, таких как WDS, Sysprep, или сторонних разработчиков.

*-Служба развертывания Windows (WDS)(развернуть образ)*

Позволяет развертывать установочные образы и специально настроенные образы на серверах и настольных компьютерах.

*-Диспетчер настроек системного центра (SCCM)(управлем настраиваем)*

Средство развертывания и управления образами систем с возможностью централизованного управления.

*-Инструментальный набор развертывания Microsoft (MDT)(делаем образ)*

Обеспечивает средства для создания образов, управления драйверами и приложениями, а также автоматизации процесса развертывания.

**23. ОС Windows: организация рабочей среды пользователя, работа с учетными записями пользователей и групп, работа с профилями пользователей.**

**Организация рабочей среды пользователя**

Рабочая среда пользователя состоит из настроек рабочего стола, например, цвета экрана, настроек мыши, размера и расположения окон, из настроек процесса обмена информацией по сети и с устройством печати, переменных среды, параметров реестра и набора доступных приложений

Для управления средой пользователя предназначены следующие средства Windows:

*Сценарий Входа в Сеть:*

Командные файлы (.bat) или исполняемые файлы (.exe), выполняемые при входе пользователя в сеть.(выполняются при кжадой регистрации пользователя в сети)

Могут включать команды для создания сетевого соединения или запуска приложений.

*Профили Пользователей:*

Хранят настройки рабочей среды пользователя.(среды компьютера,которые определены пользователем)

*Сервер Сценариев Windows (WSH):*

Независим от языка и предназначен для работы на 32-разрядных платформах Windows. Включает в себя ядро сценариев VBScript и JScript. Позволяет выполнение сценариев на рабочем столе Windows или в консоли команд. Большинство настроек пользовательской среды управляются с помощью групповых политик.(выполнение сценариев прямо на раьочем столе или на консоли)

**Работа с учетными записями пользователей и групп**

Локальные учетные записи пользователей определяются локально на устройстве и могут назначаться только на этом устройстве

Учетные записи локальных пользователей по умолчанию — это встроенные учетные записи, которые создаются автоматически при установке операционной системы.

Учетная запись локального администратора по умолчанию — это учетная запись пользователя для системного администрирования.

Учетная запись администратора имеет полный контроль над файлами, каталогами, службами и другими ресурсами на локальном устройстве. Учетная запись администратора может создавать других локальных пользователей, назначать права пользователей и назначать разрешения. Учетная запись администратора может в любое время управлять локальными ресурсами, изменив права и разрешения пользователя.

Гостевая учетная запись позволяет случайным или разовым пользователям, у которых нет учетной записи на компьютере, временно войти на локальный сервер или клиентский компьютер с ограниченными правами пользователя.

Учетные записи локальных пользователей по умолчанию и созданные вами учетные записи локальных пользователей находятся в папке «Пользователи».

**Работа с профилями пользователей(содержит базовые настройки и парамеры кторые получает каждый пользовталь)**

Система создает профиль пользователя при первом входе пользователя на компьютер. При последующих входах система загружает профиль пользователя, а затем другие системные компоненты настраивают среду пользователя в соответствии с информацией в профиле

*Типы профилей пользователей:*

Локальные профили: Хранятся на локальном жестком диске компьютера.

Перемещаемые профили: Копия локального профиля на сервере, загружается при входе в сеть.

Обязательные профили: Используются администраторами для указания параметров, изменения не сохраняются.

Временные профили: Выдаются при ошибке загрузки профиля, удаляются после каждого сеанса.

*Элементы профиля пользователя:*

Куст реестра: Файл NTuser.dat, загружается при входе, содержит настройки и конфигурацию реестра.

Папки профилей в файловой системе: Хранят файлы профиля, такие как документы, настройки приложений.

*Преимущества профилей пользователей:*

Сохранение параметров между входами пользователя.

Настроенный рабочий стол для каждого пользователя при совместном использовании компьютера.

Уникальность параметров в профиле для каждого пользователя.

Изменения в профиле одного пользователя не влияют на других пользователей.

**24. Системный реестр ОС Windows, его назначение и использование.**

Реестр — это база данных, в которой Windows хранит параметры конфигурации пользователя и компьютера. Всякий раз, когда вы вносите изменения в конфигурацию Windows, это изменение фиксируется в реестре.

Реестр Windows организован иерархически. На верхнем уровне имеется пять кустов реестра, которые представляют собой отдельный набор связанных параметров, структурированных как серия ключей, подразделов и значений: HKEY\_CLASSES\_ROOT, HKEY\_CURRENT\_USER, HKEY\_LOCAL\_MACHINE, HKEY\_USERS и HKEY\_CURRENT\_CONFIG.

HKEY\_CLASSES\_ROOT: Ассоциации файлов и приложений для открытия файлов.

HKEY\_CURRENT\_USER (HKCU): Информация о конфигурации текущего пользователя.

HKEY\_LOCAL\_MACHINE (HKLM): Важные параметры конфигурации компьютера.

HKEY\_USERS: Информация о конфигурации всех пользователей, включая текущего.

HKEY\_CURRENT\_CONFIG: Информация о текущем профиле оборудования.

Для поддержания структуры базы данных аналогичные настройки хранятся в папках и подпапках, известных как ключи и подразделы. Это упрощает ссылку на определенное значение реестра. Вы можете указать путь, объявив соответствующий куст, ключ, подразделы и значение.  


REG\_BINARY. Необработанные двоичные данные. Эти значения обычно отображаются в шестнадцатеричном формате. Информация об оборудовании часто хранится в значениях REG\_BINARY.

REG\_DWORD. 4-байтовые числа (32-битное целое число). Многие значения, связанные с драйверами устройств и службами, хранятся в значениях REG\_DWORD.

REG\_SZ. Текстовая строка фиксированной длины.

REG\_EXPAND\_SZ. Текстовая строка переменной длины.

REG\_MULTI\_SZ. Несколько строковых значений.

**25. ОС Windows: планирование и назначение разрешений NTFS.**Идентификатор безопасности (SID) — это уникальное значение переменной длины, используемое для идентификации доверенного лица. Каждая учетная запись имеет уникальный идентификатор безопасности, выданный центром сертификации, таким как контроллер домена Windows, и хранящийся в базе данных безопасности. Каждый раз, когда пользователь входит в систему, система получает идентификатор безопасности для этого пользователя из базы данных и помещает его в маркер доступа для этого пользователя. Система использует SID в маркере доступа для идентификации пользователя во всех последующих взаимодействиях с безопасностью Windows

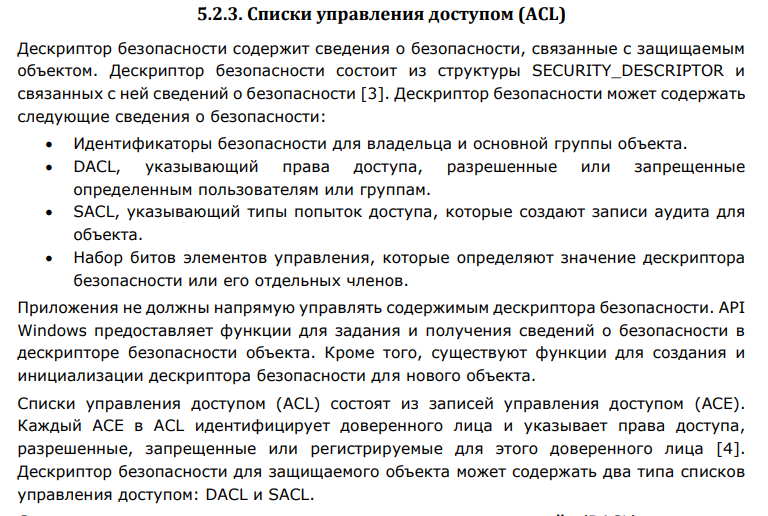
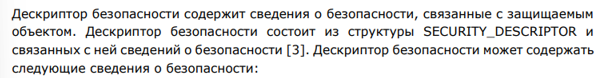
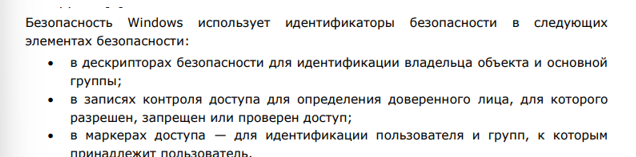
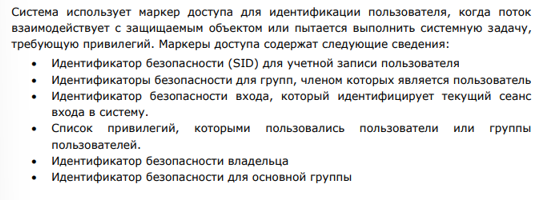
*1. Введение:*

NTFS (New Technology File System) - файловая система, используемая в операционных системах Windows.

Разрешения NTFS определяют доступ к файлам и папкам на уровне файловой системы.

*2. Понимание Разрешений:*

Разрешения определяют, кто может что делать с файлами и папками (чтение, запись, исполнение).

Каждый объект (файл или папка) имеет ACL (Access Control List), содержащий список разрешений.  
  
  
  
  


*3. Пользователи и Группы:*

Разрешения могут быть назначены пользователям или группам пользователей.

Группы упрощают управление, так как разрешения могут быть назначены целой группе.

*4. Права и Роли:*

Права: Определяют общие действия, такие как изменение системного времени.

Роли: Задают специфические задачи, такие как администратор или пользователь.

*5. Процесс Назначения Разрешений:*

Выбор Объекта: Выберите файл или папку, для которой вы хотите настроить разрешения.

Свойства и Закладка "Безопасность": Правый клик по объекту, выберите "Свойства" и перейдите на закладку "Безопасность".

Выбор Пользователя или Группы: Нажмите "Изменить" и выберите нужного пользователя или группу.

*6. Работа с Разрешениями:*

Разрешения по Умолчанию: Определяют доступ к новым объектам до изменения разрешений.

Наследование Разрешений: Разрешения могут быть унаследованы от родительских объектов.

*7. Основные Разрешения:*

Полный доступ (Full Control): Полные права на чтение, запись, изменение, удаление и выполнение.

Чтение и Исполнение (Read & Execute): Позволяет просматривать и выполнять файлы, но не изменять.

Запись (Write): Позволяет создавать и изменять файлы, но не удалять или исполнять.

*8. Пример Назначения Разрешений:*

Задача: Назначить группе "Редакторы" разрешение на изменение файлов в папке "Проекты".

Действия:

Выберите папку "Проекты".

Перейдите на закладку "Безопасность".

Нажмите "Изменить" и добавьте группу "Редакторы" с разрешением "Изменение".

*9. Проверка Разрешений:*

Проверяйте разрешения, удостоверьтесь, что необходимые группы и пользователи имеют доступ.

*10. Завершение:*

Разрешения NTFS важны для безопасности и управления файлами в Windows.

Регулярно аудитите разрешения для поддержания актуальности и безопасности.

**26. Средства автоматической настройки в ОС Windows.**

* Командное консольное окно.
* BAT-файлы.
* PowerShell.
* WMI.
* ETW.
* Групповые политики.
* Административные шаблоны.
* Файловые системы.
* Динамические диски.
* Управление сжатием на дисках NTFS.
* Шифрование файлов.
* Управление дисковыми квотами.
* Использование Консоли восстановления.

**Командное консольное окно**

В Windows есть две оболочки командной строки: командная оболочка **cmd** и **PowerShell**.

Каждая оболочка — это программная программа, которая обеспечивает прямую связь между оператором и операционной системой или приложением, предоставляя среду для автоматизации ИТ-операций.

Командная оболочка **cmd** была первой оболочкой, встроенной в Windows, для

автоматизации повседневных задач, с пакетными (.bat или .cmd) файлами.

**PowerShell** был разработан для расширения возможностей командной оболочки для выполнения команд PowerShell, называемых **командлетами**. Командлеты похожи на команды Windows, но предоставляют более расширяемый язык сценариев.

**BAT-файлы**

BAT-файл – текстовый файл, содержащий команды системы, которые могут выполняться последовательно.

Комментарии начинаются с REM (remark)

Строки, начинающиеся с @(лигатура), не выводятся на экран.

Допускаются программные конструкции, например:

**for** %%i **in** (1 1 5) **do** *echo* %%i (%% переменные в циклах, а % переменные вне цикла) ( **in** (1-нач знач 1-шаг 5-конечное знач)

**if** a *equ* 0 *echo* 0 **else** *echo* notzero (*equ- =)*

Для вызова другого скрипта используется команда call

Для создания переменной используется команда set (set a=0)

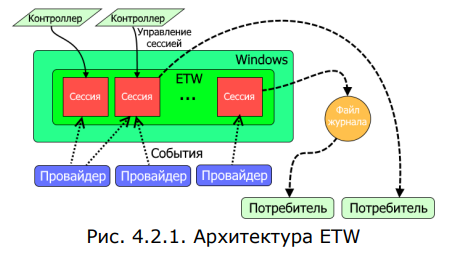
**PowerShell** — расширяемое средство автоматизации от Microsoft с открытым исходным кодом.

Инструментарий управления Windows **(WMI)** — (**универсальный переводчик (позволяет просмтраивать и управлять всеми настройками приложений**))это инфраструктура для управления данными и операциями в операционных системах Windows. Он представляет собой реализацию стандарта WBEM, ориентированного на создание универсального интерфейса мониторинга и управления различными системами.

**Event Tracing for Windows (ETW)** — это служба, которая позволяет получать события от одного или нескольких поставщиков событий в режиме реального времени или из файла \*.etl за некоторый временной период. (механизм трассировки и регистрации событий, которые генерируются приложениями в пользовательском режиме и драйверами в ядре)

Архитектура ETW включает в себя 4 элемента (рис. 4.2.1):

* поставщики событий (providers)(отслеживание событий )
* потребители событий (consumers)(это приложение обрабатывающие событие )
* контроллеры ETW (controllers)(отвечает за контроль и функционирований сессий \местоположение )
* сессии ETW (event tracing sessions)(сеанс отслеживаниия событий)



**Групповая политика** — важный элемент любой среды Microsoft Active Directory (AD). Её основная **цель** — дать ИТ-администраторам возможность централизованно управлять пользователями и компьютерами в домене.

Групповая политика **состоит из** набора политик, называемых **объектами групповой политики (GPO)**.

При создании домена AD автоматически создаются два объекта групповой политики:

• **политика домена** по умолчанию устанавливает базовые параметры для всех пользователей и компьютеров в домене в трех плоскостях: политика паролей, политика блокировки учетных записей и политика Kerberos; (все пользователи и компьютеры в домене будут следовать этим параметрам если не указано иное )

• **политика контроллеров домена** по умолчанию устанавливает базовые параметры безопасности и аудита для всех контроллеров домена в рамках домена.

Объект групповой политики не действует, пока не будет связан с контейнером Active

Directory, например, сайтом, доменом или подразделением.(МЫ ДОЛЖНЫ СВЯЗАТЬ С КОНТЕЙНЕРОМ ЧТОБЫ ПОЛИТИКА РАБОТАЛА) (что все контроллеры домена будут следовать этим параметрам безопасности и аудита)

**Административные** шаблоны (файлы) в формате XML, написанные на различных языках, которые задают параметры групповых политик, основанные на значениях реестра. Административные шаблоны позволяют администраторам устанавливать и изменять настройки безопасности, поведения и другие параметры на компьютере или сети через групповые политики.

**Два вида административных шаблонов:**

* **ADMX** — не зависящий от языка файл установки, который указывает количество и тип параметров политики, расположение по категориям согласно отображению файла в редакторе локальных групповых политик.(не зависят от языка \содержат все настройки которые вы можете опрпделить \определяют количество и тип политик)
* **ADML** — файл установки на определенном языке, который предоставляет

связанные с языком сведения для ADMX-файла. позволяет параметру политики отображаться в редакторе локальных групповых политик на нужном языке. Вы можете добавлять новые языки, добавляя новые ADML-файлы на нужных языках.(содержит сведения для ADMX файла \позволяют параметру политики локальных грцпповых политик отображаться в редакторе на нужном языке )  
Таким образом, основное различие между ADMX и ADML заключается в том, что ADMX определяет, какие политики доступны для настройки, а ADML определяет, как эти политики отображаются и описываются на разных языках

**27. Основные понятия системы UNIX. Пользователи системы, атрибуты пользователя. Файловая структура ОС.** «Философия UNIX гласит:

 Пишите программы, которые делают что-то одно и делают это хорошо.

Пишите программы, которые бы работали вместе.

Пишите программы, которые бы поддерживали текстовые потоки, поскольку это универсальный интерфейс» **Unix** - это семейство операционных систем (ОС), обладающих сходной архитектурой и интерфейсом с пользователем.

**Основные современные варианты** UNIX: Linux, BSD (FreeBSD, NetBSD, OpenBSD), AIX, HPUX, Solaris, SCO.

**Важнейшие стандарты**, обеспечивающие целостность семейства UNIX:

**POSIX** - Portable Operating System Interface  
 **ANSI С** (c89 и c99)

Классическая архитектура UNIX двухуровневая

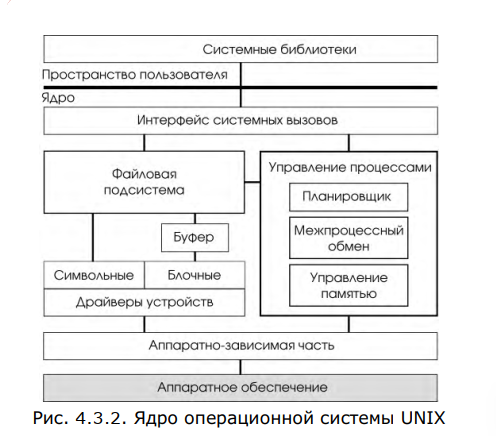
**Ядро** — управляет ресурсами компьютера и предлагает программам базовый набор услуг (системные вызовы).

**Системные программы** (управление сетью, терминалами, печатью), прикладные программы (редакторы, утилиты, компиляторы и т.д.).

Операционная система UNIX обладает классическим монолитным ядром, в котором можно

выделить следующие основные части: файловая подсистема, управление процессами и

драйверы устройств.



Функции ядра:

1. Инициализация системы - загрузка и запуск ОС.
2. Управление процессами и потоками.
3. Управление памятью - отображение адресного пространства на физическую память, совместное использование памяти процессами.
4. Управление файлами - реализует понятие файловой системы, дерева каталогов и файлов.
5. Обмен данными между процессами, выполняющимися внутри одного компьютера, в разных узлах сетей передачи данных, а также между процессами и драйверами внешних устройств.
6. Программный интерфейс (API) - обеспечивает доступ к возможностям ядра со стороны процессов пользователя через системные вызовы, оформленных в виде библиотеки функций на Си.

Ядро изолирует программы пользователя от аппаратуры. Все части системы, не считая небольшой части ядра, полностью независимы от архитектуры компьютера и написаны на Си. Системные вызовы - это уровень, скрывающий особенности конкретного механизма выполнения на уровне аппаратуры от программ пользователя. Для программиста, системный вызов - это функция (определенная на Си), которую он вызывает в своей программе. Все низкоуровневые операции осуществляются через системные вызовы.

**Оболочка (shell)** - это программа, которая позволяет вам связываться с операционной

системой. Она считывает команды, которые вы вводите, и интерпретирует их как

запросы на выполнение других программ, на доступ к файлу или обеспечение вывода.

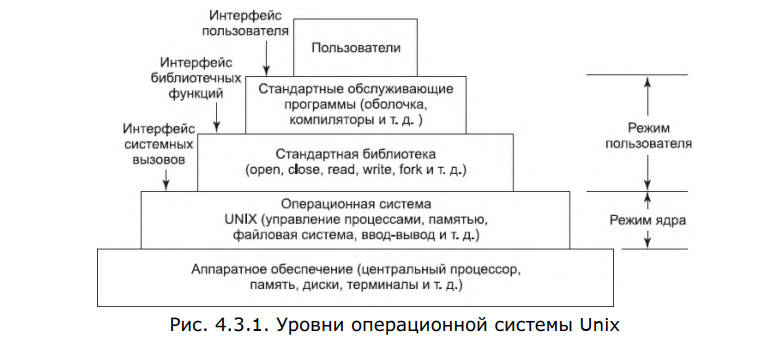
Оболочка также поддерживает мощный язык программирования, который обеспечивает

условное выполнение и управление потоками данных.

ДАЛЕЕ ИДЕТ ПОДРОБНАЯ ИНФОРМАЦИЯ.

Существует два основных объекта операционной системы UNIX, с которыми приходится

работать пользователю – файлы и процессы. Они определяют архитектуру системы

**Операционная система работает на «голом железе». Ее функция заключается в управлении аппаратным обеспечением и предоставлении всем программам интерфейса системных вызовов. Эти системные вызовы позволяют программам пользователя создавать процессы, файлы и прочие ресурсы, а также управлять ими. Всю систему можно рассматривать как пирамиду   
**

**Пользователи системы, атрибуты пользователя**

Прежде чем клиент сможет начать работу с ОС UNIX, он должен стать пользователем

системы, т.е. получить имя, пароль и ряд других атрибутов.

 Пользователем является объект, который обладает определенными правами и может запускать на выполнение программы и владеть файлами.

В системе существует один пользователь, обладающий неограниченными правами это суперпользователь или администратор системы (обычно с именем root)

UID - УНИКАЛЬНЫЙ ИДЕНТИФИКАТОР ПОЛЬЗОВАТЕЛЯ.

Группа — список пользователей, имеющих сходные задачи

GID – Каждая группа имеет уникальное имя, а система различает группы по групповому

идентификатору (GID)Информация о пользователях обычно хранится в

специальном файле: /etc/passwd, о группах — /etc/group. Этот файл доступен только

для чтения. Писать в него может только администратор.

Каждая строка файла  /etc/passwd является записью конкретного пользователя и имеет следующий

формат:

name:passwd-encod:UID:GID:comments:home-dir:shell

— всего семь полей (атрибутов), разделенных двоеточиями.

Рассмотрим подробнее каждый из атрибутов.

**name** Регистрационное имя пользователя

**passwd-encod** Пароль пользователя в закодированном виде. Алгоритмы кодирования известны, но они  не позволяют декодировать пароль.

**UID** Идентификатор пользователя является внутренним представлением пользователя в

системе. Этот идентификатор наследуется задачами, которые запускает пользователь, и

файлами, которые он создает.Суперпользователь имеет UID=0, что дает ему неограниченные права в системе  
**GID** Определяет идентификатор первичной группы пользователя. Этот идентификатор

соответствует идентификатору в файле /etc/group, который содержит имя группы и

полный список пользователей, являющихся ее членами. Принадлежность пользователя

к группе определяет дополнительные права в системе. Группа определяет общие для

всех членов права доступа и тем самым обеспечивает возможность совместной работы

(например, совместного использования файлов).

**comments** Обычно, это полное "реальное" имя пользователя. Это поле может содержать

дополнительную информацию, например, телефон или адрес электронной почты.

**home-dir** Домашний каталог пользователя. При входе в систему пользователь оказывается в этом каталоге. Как правило, пользователь имеет ограниченные права в других частях

файловой системы, но домашний каталог и его подкаталоги определяют область

файловой системы, где он является полноправным хозяином.

**shell** Имя программы, которую UNIX использует в качестве командного интерпретатора. При

входе пользователя в систему UNIX автоматически запустит указанную программу.

Обычно это один из стандартных командных интерпретаторов /bin/sh (Bourne shell), /bin/csh (С shell) или /bin/ksh (Korn shell),

**Файловая структура ОС.** почему-то в данном разделе рассматривается всё на примере LINUX

Файл в системе Linux — это последовательность байтов произвольной длины (от 0 до

некоторого максимума), содержащая произвольную информацию. Не делается различия

между текстовыми (ASCII) файлами, двоичными файлами и любыми другими типами

файлов. Значение битов в файле целиком определяется владельцем файла. Системе это

безразлично. Имена файлов ограничены 255 символами. Всё зависит от системы.   
**1ой была система MINIX1** имена файлов были ограничены в ней 14 символами максимальный размер файла составлял 64 Мбайт  
**2ой была ext,** которая позволяла использовать имена файлов длиной 255 символов и размер файлов 2 Гбайт (однако она была медленнее, чем файловая система MINIX 1)

**3ей была ext2** (с длинными именами файлов, большими файлами и более высокой производительностью),  которая и стала основной файловой системой.  
но линукс поддерживает большое множество файловых систем.

По соглашению многие программы ожидают, что имена файлов будут состоять из

основного имени и расширения, разделенных точкой (которая также считается

символом). Так, prog.c — это обычно программа на языке C

Система содержит каталоги, подкаталоги и фалы в них. Символ / разделяет их в пути

Таблица 4.3.1. Некоторые важные каталоги, существующие в большинстве систем Linux

|  |  |
| --- | --- |
| Каталог | Содержание |
| bin | Двоичные (исполняемые) программы |
| dev | Специальные файлы для устройств ввода-вывода |
| etc | Разные системные файлы |
| lib | Библиотеки |
| usr | Каталоги пользователей |

Расскажите про АБСОЛЮТНЫЙ и ОТНОСИТЕЛЬНЫЙ путь (путь относительно рабочего каталога ./)  .Это  2 способа указать путь к файлу

Для **безопасного совместного доступа** к файлам стандарт POSIX предоставляет гибкий и детальный механизм, позволяющий процессам за одну неделимую операцию **блокировать даже единственный байт файла** (или целый файл). Механизм блокировки требует от вызывающей стороны указать блокируемый файл, **начальный бай**т и **количество байтов**. Если операция завершается успешно, то система создает запись в таблице, в которой указывается, что определенные байты файла (например, запись базы данных) заблокированы. Стандартом определены два типа блокировки**: блокировка с монополизацией** (exclusive locks) и блокировка **без монополизации** (shared locks). Если часть файла уже имеет блокировку без монополизации, то повторная попытка установки блокировки без монополизации на это место файла разрешается, но попытка установить блокировку с монополизацией будет отвергнута. Если же какая-либо область файла содержит блокировку с монополизацией, то любые попытки заблокировать любую часть этой области файла будут отвергаться, пока не будет снята блокировка. Для успешной установки блокировки необходимо, чтобы каждый байт в блокируемой области был доступен.

**Расположение системных файлов и программ**

Файлы ядра можно обнаружить в каталоге **/boot**

Одна из наиболее распространенных директорий для установки программ — это **/usr/bin**

**/opt**. Она служит для установки приложений, которые не являются частью основной операционной системы. Здесь можно разместить программы, которые вы загрузили из Интернета или с которыми не связаны зависимости от других программ.

важно упомянуть о **/usr/local** — директории, в которой обычно размещаются

локально установленные программы. Здесь могут находиться программы, установленные

самим пользователем или администратором системы и которые не входят в стандартный

пакет установки Linux.  **--------------------------------------------------------------------------------------------**

ОС UNIX базируется на работе с двумя основными объектами - **файлами** и **процессами**.

**Файлы** хранят данные пользователя, а **процессы** определяют функциональность системы.

Работа с файлами осуществляется через чтение и запись специальных файлов, а выполнение программ включает считывание исполняемого кода из файла в память.

Операционная система UNIX организована как пирамида, где аппаратное обеспечение находится в основании, а операционная система управляет аппаратурой и предоставляет интерфейс системных вызовов для программ.

Программы используют системные вызовы для создания и управления процессами, файлами и другими ресурсами.

Графический интерфейс пользователя основан на оконной системе X, которая управляет визуальным представлением на экране.

Операционная система также включает стандартные программы, такие как командный процессор, компиляторы, редакторы, и поддерживает графические интерфейсы пользователя через X Windowing System.

**Пользователи системы**

Прежде чем клиент сможет начать работу с ОС UNIX, он должен стать пользователем системы, т.е. получить имя, пароль и ряд других атрибутов. **Пользователем** является объект, который обладает определенными правами и может запускать на выполнение программы и владеть файлами. Пользователями могут быть отдельные клиенты, удаленные компьютеры или группы пользователей с одинаковыми правами и функциями. В системе существует один пользователь, обладающий неограниченными правами это **суперпользователь** или **администратор** системы (обычно с именем **root**).

Система различает пользователей по идентификатору пользователя UID.

**Группа** — список пользователей, имеющих сходные задачи. Каждая группа имеет уникальное имя, а система различает группы по групповому идентификатору (GID). Информация о пользователях обычно хранится в специальном файле: */etc/passwd,* о группах *— /etc/group*. (файлы доступны только для чтения, писать может только админ)

**Атрибуты пользователя**

все атрибуты пользователя хранятся в файле /etc/passwd

Каждая строка файла является записью конкретного пользователя и имеет следующий формат:

name:passwd-encod:UID:GID:comments:home-dir:shell

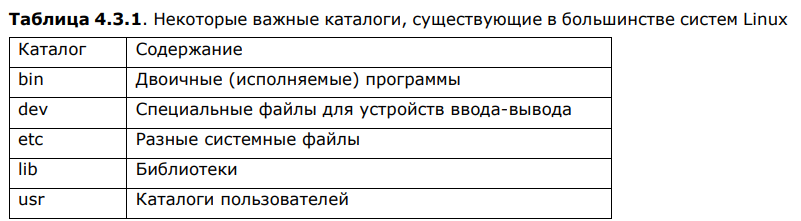
всего семь полей (атрибутов), разделенных двоеточиями.

* name: Регистрационное имя пользователя, используемое при входе в систему.
* passwd-encod: Закодированный пароль пользователя. Алгоритм кодирования предотвращает декодирование пароля. В некоторых системах пароль хранится в отдельном файле, а в поле passwd-encod может быть символ 'х' или '!'.
* UID: Идентификатор пользователя, внутреннее представление пользователя в системе. Суперпользователь (администратор) имеет UID=0.
* GID: Идентификатор первичной группы пользователя, соответствует идентификатору в файле /etc/group.
* comments: Полное "реальное" имя пользователя или дополнительная информация.
* home-dir: Домашний каталог пользователя, место, где пользователь оказывается при входе в систему.
* shell: Имя программы, используемой в качестве командного интерпретатора при входе пользователя. Обычно это /bin/sh, /bin/csh или /bin/ksh.

**Файловая структура ОС.**

**Файл в системе Linux** — это последовательность байтов произвольной длины (от 0 до некоторого максимума), содержащая произвольную информацию. Не делается различия между текстовыми (ASCII) файлами, двоичными файлами и любыми другими типами файлов. Значение битов в файле целиком определяется владельцем файла. Имена файлов ограничены 255 символами. В именах файлов разрешается использовать все ASCII-символы, кроме символа NULL.

Для удобства файлы могут группироваться в **каталоги**. Каталоги хранятся на диске в виде файлов. Каталоги могут содержать подкаталоги, что приводит к иерархической файловой системе. Корневой каталог называется **/**



Существует два способа указания имени файла в системе Linux: абсолютный путь, начинающийся от корневого каталога, и относительный путь, указывающий относительно текущего рабочего каталога.

Стандарт POSIX обеспечивает гибкий механизм блокировки файлов, позволяя процессам блокировать как единичные байты, так и целые файлы за одну неделимую операцию. Стандарт определяет два типа блокировок: с монополизацией (эксклюзивные) и без монополизации (общие). Если часть файла имеет блокировку без монополизации, то можно повторно установить такую же блокировку, но блокировка с монополизацией будет отвергнута. Если область файла заблокирована с монополизацией, то любые попытки блокировки этой области будут отвергнуты до снятия блокировки. Успешная установка блокировки требует доступности каждого байта в заблокированной области.

**28. ОС UNIX: особенности процессов, сигналы, обработка сигналов**.

Основными активными сущностями в системе Linux являются **процессы**. Каждый процесс выполняет одну программу и изначально получает один поток управления. Иначе говоря, у процесса есть один счетчик команд, который отслеживает следующую исполняемую команду. Linux позволяет процессу создавать дополнительные потоки (после того, как он начинает выполнение).  
Основными активными сущностями в системе являются **процессы**.

Каждый **процесс** выполняет **одну программу** и изначально получает **один поток**

управления. Иначе говоря, у процесса есть один счетчик команд, который отслеживает

следующую исполняемую команду. Linux позволяет процессу создавать дополнительные

потоки   
Каждый процесс выполняет одну программу и изначально получает один поток

управления. Иначе говоря, у процесса есть один счетчик команд, который отслеживает

следующую исполняемую команду. Linux позволяет процессу создавать дополнительные

потоки работают десятки фоновых процессов, называемых **демонами** (**daemons**). Они

запускаются при загрузке системы из сценария оболочк  
**Типичным демоном является cron** Он просыпается раз в минуту, проверяя, не нужно ли

ему что-то сделать. Если у него есть работа, он ее выполняет, а затем отправляется спать

дальше (до следующей проверки). Этот демон позволяет планировать в системе Linux

активность на минуты, часы, дни и даже месяцы вперед

**FORK -** создает точную копию исходного процесса, называемого родительским

процессом (parent process). Новый процесс называется дочерним процессом (child

process). У родительского и у дочернего процессов есть собственные (приватные) образы

памяти. Если родительский процесс впоследствии изменяет какие-либо свои

переменные, то эти изменения остаются невидимыми для дочернего процесса (и

наоборот

Системный вызов **fork** создает точную копию исходного процесса, называемого родительским процессом (parent process). У родительского и у дочернего процессов есть собственные (приватные) образы памяти. Если родительский процесс впоследствии изменяет какие-либо свои переменные, то эти изменения остаются невидимыми для дочернего процесса (и наоборот).

**fork** возвращает дочернему процессу число 0, а родительскому — отличный от нуля PID дочернего процесса. **getpid**

Типы процессоров:

**Системные процессы** - часть ядра, всегда в оперативной памяти. Не имеют соответствующих исполняемых файлов, запускаются особым образом при инициализации ядра. Примеры: shed, vhand, bdfflush, kmadaemon, и init.

**Демоны (daemons)** фоновые процессы, работающие в фоновом режиме и обслуживающие различные системные задачи или услуги. Они часто запускаются при загрузке системы и работают независимо от активности конкретного пользователя. Типичным демоном является ***cron***. Он просыпается раз в минуту, проверяя, не нужно ли ему что-то сделать. Есть работа-выполняет, нету-спит до след проверки. Позволяет планировать в системе Linux активность на минуты, часы, дни и месяцы вперед.

**Прикладные процессы** - все остальные, выполняющиеся в системе, часто порождаемые пользовательским сеансом. Время их выполнения ограничено сеансом работы.

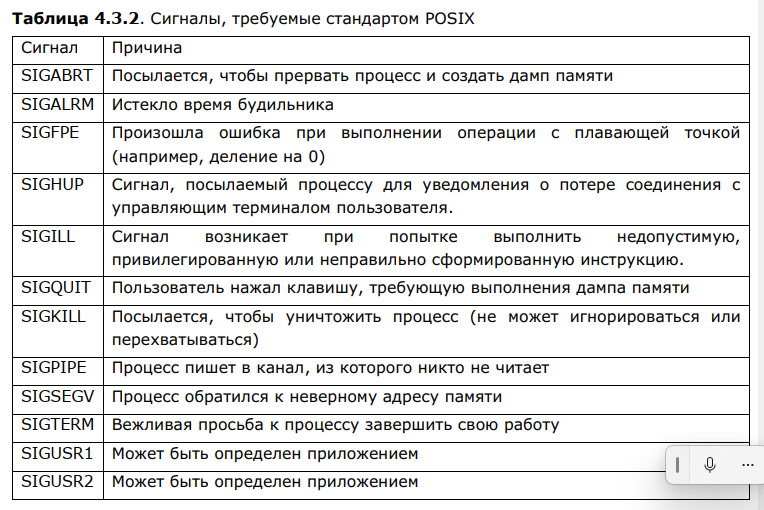
**Сигналы** — это способ информирования процесса со стороны ядра о происшествии некоторого события. Один процесс может посылать сигнал другому через системный вызов kill.

**Обработка сигналов** в Linux осуществляется через установку обработчиков сигналов. Когда процесс получает сигнал, ядро отправляет управление к обработчику, который был предварительно установлен для этого сигнала. Процесс может выбрать различные действия при получении сигнала:

**Игнорирование (SIG\_IGN):** Процесс может проигнорировать сигнал, если установлен обработчик SIG\_IGN (ignore).

**По умолчанию (SIG\_DFL)**: Если не установлен пользовательский обработчик, применяется действие по умолчанию (default). Например завершение процесса.

**Пользовательский обработчик (функция):** Процесс может установить свою собственную функцию-обработчик. Когда сигнал поступает, управление передается этой функции.

**сигналы являются программными прерываниями** [1]. Один процесс может послать другому так называемый сигнал (signal). Процессы могут сообщить системе,какие действия следует предпринимать, когда придет входящий сигнал. **Варианты такие:проигнорировать сигнал, перехватить его, позволить сигналу убить процесс (действие** **по умолчанию для большинства сигналов).**

**29. Основные принципы функционирования Linux. Основные компоненты Linux. Дистрибутивы Linux. Файловая система Linux**.   
**клиент-сервер**

Каждый **пользователь** работает с системой через виртуальный **терминал**, которых может быть до 12-ти

**существует возможность изменять существующие интерфейсы и создавать свои собственные**

**В ОС Linux все файлы организованы в непрерывный поток байтов.** Данные, вводимые с

клавиатуры, представляют собой входной поток данных, а отображаемые данные –

выходной поток.они могут свободно взаимодействовать с файлами  монитор и клавиатура рассматриваются системой как файлы

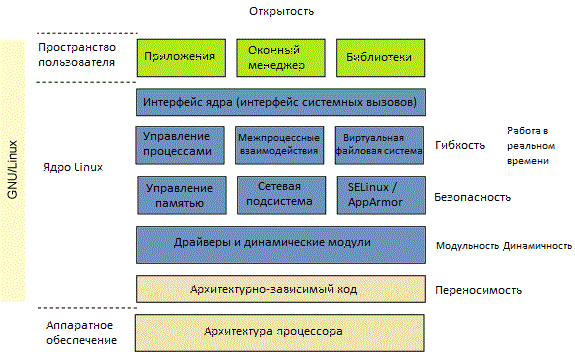
**Файловая система** в ОС Linux как и в большинстве других систем имеет **иерархическую**

(древовидную) структуру. Вверху дерева всегда находится корневой каталог **ROOT**

Linux организован по принципу клиент-сервер, где компьютеры в сети взаимодействуют, создавая единый компьютер. Пользователи работают через виртуальные терминалы. Один компьютер может одновременно выполнять роль сервера и рабочей станции.

Linux - это ядро и ядро каждого дистрибутива Linux. Программное обеспечение ядра Linux поддерживается группой людей, возглавляемой Линусом Торвальдсом.

ОС Linux обладает различными компонентами и возможностями:

* *Планировщик*: Управляет приоритетами процессов.
* *Файл подкачки:* Используется для хранения неактивных процессов на диске при переполнении оперативной памяти.
* *Модули:* Позволяют динамически добавлять поддержку новых устройств, что помогает сократить размер ядра.
* *Файловые системы:* Поддерживают различные типы файловых систем, включая ext3, ReiserFS, VFAT и NTFS.
* *Механизмы защиты:* Используются для установки разрешений на чтение, запись и выполнение файлов, обеспечивая безопасность.
* *Инструменты администрирования*: управления пользователями, дисками, сетями и другими ресурсами.
* *Серверные возможности:* Позволяют использовать Linux в роли сервера, предоставляя различные службы, такие как веб-серверы, почтовые серверы и другие.
* *Система управления пакетами:* Позволяет автоматизировать установку, обновление и удаление программного обеспечения, используя различные форматы пакетов, такие как RPM, DEB, и другие.  
  

Мы можем распределить **дистрибутивы Linux** на три группы:

***Enterprise Grade Linux*** Дистрибутивы этой группы предназначены для развертывания в крупных организациях с использованием оборудования предприятия.( Red Hat Enterprise Linux, CentOS, SUSE Linux Enterprise Server, Debian GNU/Linux , Ubuntu LTS)

Особенности:

* Стабильность и долгосрочная поддержка.
* Ориентированы на надежность и безопасность.
* Медленное внедрение новых технологий.
* Предоставляют более старые, но стабильные версии программного обеспечения.

***Consumer Grade Linux*** Дистрибутивы этой группы больше ориентированы на малый бизнес или домашних пользователей и любителей.( Fedora, Ubuntu non-LTS, openSUSE)

Особенности:

* Ориентированы на новое оборудование и современные драйверы.
* Регулярные выпуски с обновлениями.
* Более быстрое внедрение новых технологий.

***Experimental and Hacker Linux*** Дистрибутивы этой группы используют самые современные технологии. (Arch, Gentoo)

Особенности:

* Использование самых последних технологий.
* Скользящая модель выпуска.
* Предоставляют возможность для тестирования будущих функций.

Разработаны для пользователей, готовых к возможным проблемам и изменениям.

**Дистрибутивы состоя**т из отдельных **пакетов**, каждый из которых содержит какое-то

приложение, утилиту или сервис. Отдельный пакет может содержать, например, веббраузер, библиотеку для работы с графическими файлами в формате PNG, набор

шрифтов и так далее.

Программное обеспечение, содержащееся в пакете, поставляется в одном из двух

основных видов:

1. в виде бинарных файлов, которые предназначены для непосредственной установки в вашу систему, без какой-либо дополнительной обработки (например, компиляции).
2. в виде исходных текстов, которые обычно содержат текст на каком-то языке программирования, заархивированный в формате tar и упакованный программой gzip, а также вспомогательные файлы, необходимые для компиляции приложения из файлов пакета.

**Файловая система** в ОС Linux как и в большинстве других систем имеет иерархическую (древовидную) структуру. Все объекты являются файлами, в том числе и директории для организации доступа к файлам.

Существуют следующие типы файлов: обычные файлы, каталоги, символьные ссылки, блочные устройства, символьные устройства, сокеты, каналы. Тип выводится **ls -l.**

всегда есть только один корневой каталог, который называется /

Абсолютные и относительные пути

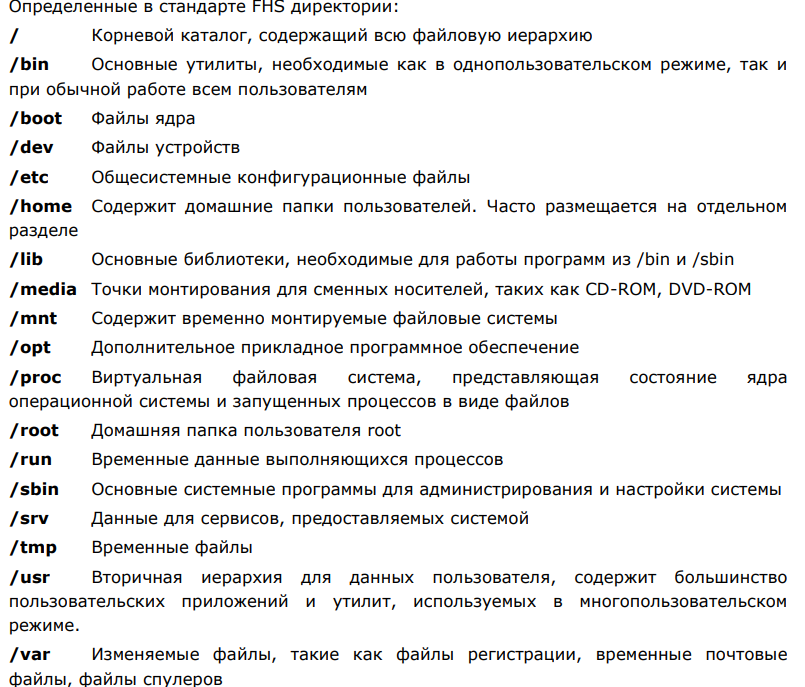
Имя может содержать любые символы (в том числе и кириллицу), кроме / ? < > \* " | Максимальная длина имени файла — 254 символа

Точка **.** в начале имени файла делает его **скрытым**, то есть, он не показывается в выводе команды ls.

Команда **pwd** (Print Working Directory) возвращает полный путь к текущей директории командной оболочки (хранится в переменной PWD) Для смены текущей директории используется команда **cd** (Change Directory)

**Домашний каталог (домашняя папка, домашняя директория)** – предназначен для хранения собственных данных пользователя Linux и личных настроек для программ. Полный путь к домашнему каталогу хранится в переменной окружения **HOME**. Для обычных пользователей домашний каталог находится в директории **/home**

**FHS (Filesystem Hierarchy Standard)** – стандарт файловой системы. В FHS все файлы и каталоги находятся внутри корневого каталога, даже если они расположены на различных физических носителях. В корневой директории должны быть следующие директории:



**30. ОС Linux: управление процессами, выполнение задач в фоновом режиме, изменение приоритетов выполняющихся программ.   
Каталог /proc –** виртуальная файловая система, его содержимое не записывается на

диск, а находится в памяти

**Динамически заполняется каждый раз при загрузке компьютера и постоянно отражает**

**текущее состояние системы**

Каталог **/proc сод**ержит информацию о:

• **выполняющихся процессах**

• **конфигурации ядра**

• **аппаратном обеспечении**

Файл **/proc/cpuinfo «хранит»** информацию о **процессоре** системы.

Файл **/proc/cmdline** хранит **строки, переданные ядру при загрузке**.

Файл /**proc/modules** показывает список **модулей,** загруженных **в ядро.**

Каталог **/proc/sys** содержит параметры **конфигурации ядра.**

Файлы распределены по категориям, для каждой категории создан свой подкаталог.

Большинство этих параметров имеют **только два значения**:

• 0 – выключено

• 1 – включено  
Управление процессами

Процессы существуют в **иерархии**: после загрузки ядра в память запускается первый

процесс (**init** или **systemd**), который, в свою очередь, запускает другие процессы,

которые, опять же, могут запускать другие процессы.

процесс **имеет** уникальный **идентификатор** (**PID**) и идентификатор **родительского**

процесса (**PPID**).

Команда **top** динамически отображает все запущенные процессы:

Команды l t m управляют отображением строк статистики. Для сортировки служат

команды M (память), N (ID процесса), T (время выполнения), P (процент загрузки

процессора). Изменение порядка сортировки – R

Команда ps выводит статическую информацию о процессах. Без опций выводит только

процессы, относящиеся к текущей оболочке.

Завершить процесс можно с помощью команды kill.

**Выполнение задач в фоновом режиме**

Для выполнения команды в **фоновом режиме** достаточно добавить в **конце** символ

**амперсанда &.**

Узнать состояние всех остановленных и выполняемых в фоновом режиме задач в рамках

текущей сессии терминала можно при помощи утилиты **jobs** c использованием **опции -l.**

Вывод содержит порядковый номер задачи, идентификатор фонового процесса,

состояние задачи и название команды, которая запустила задание.

В любое время можно вернуть процесс из фонового режима на передний план. Для этого

служит команда fg. Если в фоновом режиме выполняется несколько программ, следует

также указывать номер. Например:

$ fg %1

Если изначально процесс был запущен обычным способом, его можно перевести в

фоновый режим, выполнив следующие действия:

• Остановить выполнение команды, нажав комбинацию клавиш Ctrl+Z.

• Перевести процесс в фоновый режим при помощи команды bg

**Изменение приоритетов выполняющихся программ**

Утилита **nice** — **программа**, предназначенная для запуска процессов с изменённым

приоритетом nice. Приоритет nice (целое число) процесса используется планировщиком

процессов ядра ОС при распределении **процессорного времени** между **процессами**

Приоритет **nice** — **число**, указывающее **планировщику процессов ядра** ОС **приоритет**,

который **пользователь** хотел бы **назначить процессу**.

Утилита **nice**, запущенная **без аргументов**, выводит приоритет nice, унаследованный от

родительского процесса. nice принимает аргумент «смещение» в д**иапазоне от -20**

**(наивысший приоритет) до +19 (низший приоритет**).

Если указать смещение и путь к исполняемому файлу, утилита nice получит приоритет своего процесса, изменит его на указанное смещение и использует системный вызов семейства exec() для замещения кода своего процесса кодом из указанного исполняемого файла. Команда nice сделает то же, но сначала выполнит системный вызов семейства fork() для запуска дочернего

процесса (sub-shell). Если смещение не указано, будет использовано смещение +10.

Привилегированный пользователь (root) может указать отрицательное смещение.

**Планировщик процессов ядра ОС Linu**x поддерживает приоритеты о**т 0 (реальное время)**

**до 139 включительно. Приоритеты -20…+19 утилиты или команды nice соответствуют**

**приоритетам 100…139 планировщика процессов.** Другие приоритеты планировщика

процессов можно установить командой **chrt** из пакета **util**-**linux**.

Посмотреть приоритет процессов можно например с помощью утилиты **top**.

Для того, чтобы изменить приоритет у существующего процесса (т.е. такого процесса,

который ранее был уже запущен), необходимо воспользоваться командой:

**$ renice [значение приоритета] -p [id процесса]**

**$ renice -n 15 -p 21349**

**21349 (process ID) old priority 0, new priority 15**

Согласно правилам, обычный пользователь может только увеличивать значение nice

(уменьшать приоритет) любого процесса.

Процессы существуют в иерархии: после загрузки ядра в память запускается первый процесс (init или systemd), который, в свою очередь, запускает другие процессы, которые, опять же, могут запускать другие процессы.

Каждый раз, когда пользователь вводит команду, запускается программа и генерируется один или несколько процессов. Каждый процесс имеет уникальный идентификатор (**PID**) и идентификатор родительского процесса (**PPID**).

Команда **top** динамически отображает все запущенные процессы

Иерархию процессов команда **pstree**.

Завершить процесс **kill**.

Для выполнения команды в **фоновом режиме** достаточно добавить в конце символ амперсанда **&**.

Работая в фоновом режиме, команда все равно продолжает выводить сообщения в терминал, из которого была запущена. Для этого она использует потоки stdout и stderr, которые можно закрыть при помощи следующего синтаксиса:

$ command > /dev/null 2>&1 &

Узнать состояние всех остановленных и выполняемых в фоновом режиме задач в рамках текущей сессии терминала можно при помощи утилиты **jobs** **-l.**

В любое время можно вернуть процесс из фонового режима на передний план **fg**. Если в фоновом режиме выполняется несколько программ, следует указывать номер. Например: **$ fg %1**

Для завершения фонового процесса применяют команду **kill** с номером программы

Утилита **nice** — программа, предназначенная для запуска процессов с изменённым приоритетом nice.

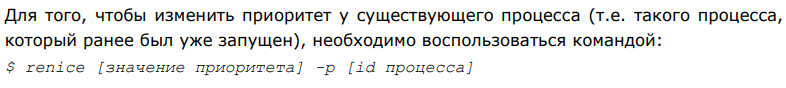
Приоритет **nice** (целое число) процесса используется планировщиком процессов ядра ОС при распределении процессорного времени между процессам

nice – значение любезности, чем меньше, тем выше приоритет

значение nice – минимальное значение приоритета =лучшее значение = самый высокий приоритет.

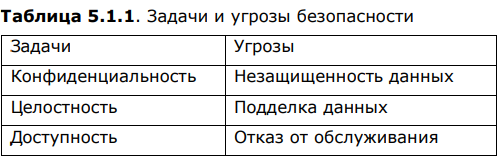
nice – диапазон приоритетов [-20, 19], default = 0;

Чтобы установить значение nice ниже нуля, требуются права суперпользователя



**31. Понятие безопасности ОС. Основные угрозы безопасности ОС. Методы и защитные механизмы операционных систем.**

**Безопасность** в контексте операционных систем охватывает механизмы и стратегии, направленные на защиту информации и обеспечение нормальной работы системы.



Первое свойство — конфиденциальность (confidentiality) — направлено на сохранение секретности данных

Второе свойство — целостность (integrity) — означает, что пользователи, не обладающие необходимыми правами, не должны иметь возможности изменять какие-либо данные без разрешения их владельцев.

Третье свойство — доступность (availability) — означает, что никто не может нарушить работу системы и вывести ее из строя

Классификация угроз **по цели** атаки:

• несанкционированное чтение информации;

• изменение информации;

• уничтожение информации;

• полное или частичное разрушение операционной системы.

Классификация угроз **по принципу воздействия** на операционную систему:

• использование известных (легальных) каналов получения информации

• использование скрытых каналов получения информации

• создание новых каналов получения информации с помощью программных закладок.

Классификация угроз **по характеру** воздействия на ОС:

• активное воздействие – несанкционированные действия злоумышленника в системе;

• пассивное воздействие – несанкционированное наблюдение злоумышленника за процессами, происходящими в системе

Операционная система **может подвергнуться следующим типичным атакам**:

• *сканирование файловой системы.* Злоумышленник просматривает файловую систему компьютера и пытается прочесть (или скопировать) все файлы подряд.

• *подбор пароля.* несколько методов подбора паролей:

* тотальный перебор;
* тотальный перебор, оптимизированный по статистике встречаемости символов или с помощью словарей;
* подбор пароля с использованием знаний о пользователе (его имени, фамилии, даты рождения, номера телефона и т. д.);

• *кража ключевой информации*. Носитель с ключевой информацией (смарткарта, Touch Memory и т. д.) может быть просто украден;

• *сборка мусора*. Во многих операционных системах удаленная пользователем информация не физически уничтожается, а лишь помечается как удаленная, что предоставляет возможность злоумышленникам восстановить и просмотреть данные.

• *превышение полномочий*. Злоумышленник, используя ошибки в программном обеспечении ОС получает полномочия, превышающие те, которые ему предоставлены в соответствии с политикой безопасности.

• *программные закладки.*

• *жадные программы* – это программы, преднамеренно захватывающие значительную часть ресурсов компьютера, в результате чего другие программы не могут выполняться или выполняются крайне медленно. Запуск жадной программы может привести к краху операционной системы.

Существуют ***два основных подхода*** *к созданию защищенных операционных систем*: **фрагментарный**, где защитные функции добавляются отдельно после создания основной системы, и **комплексный**, где защитные функции внедряются в архитектуру системы на этапе проектирования, обеспечивая их тесное взаимодействие и интеграцию.

Основные административные меры защит.

• Постоянный контроль корректности функционирования ОС, особенно ее подсистемы защиты

• Организация и поддержание адекватной политики безопасности.

•Осведомление пользователей о необходимости соблюдения мер безопасности при работе с ОС и контроль за соблюдением этих мер.

• Регулярное создание и обновление резервных копий программ и данных ОС.

• Постоянный контроль изменений в конфигурационных данных и политике безопасности ОС.  
**Оптимальная адекватная политика безопасности** – это такая политика

безопасности, которая не только не позволяет злоумышленникам выполнять

несанкционированные действия, но и не приводит к описанным выше негативным

эффектам.

1 Анализ угроз.

2 Формирование требований к политике безопасности.

3 Формальное определение политики безопасности.

4 Претворение в жизнь политики безопасности.

5 Поддержание и коррекция политики безопасности

**Архитектура подсистемы защиты операционной системы**

1 Идентификация и аутентификация

2 Разграничение доступа.

**3 Аудит**. Операционная система регистрирует в специальном журнале события,

потенциально опасные для поддержания безопасности системы.

4 Управление политикой безопасности.

5 Криптографические функции.

6 Сетевые функции.

**Авторизация и аутентификация пользователей**

**Идентификация** субъекта доступа заключается в том, что пользователь (субъект)

сообщает операционной системе идентифицирующую информацию о себе (имя, учетный

номер) и таким образом идентифицирует себя.

**Аутентификация** субъекта доступа заключается в том, что субъект предоставляет

операционной системе, помимо идентифицирующей информации, еще и

аутентифицирующую информацию, подтверждающую, что он действительно является

тем субъектом доступа, к которому относится идентифицирующая информация.

**Авторизация** субъекта доступа происходит после успешной идентификации и аутентификации. При авторизации субъекта ОС выполняет действия, необходимые для того, чтобы субъект мог начать работу в системе.

**Криптография** преследует **четыре** основных цели [3].

**Конфиденциальность** – предоставляет доступ к информации только

авторизованным пользователям.

**Целостность** – гарантирует, что над информацией не производились

манипуляции.

**Подлинность** – подтверждает подлинность информации или личность

пользователя.

**Обеспечение невозможности отказа** – лишает возможности отрицать прежние

обязательства или действия.

**Алгоритм шифрования** - это процедура, которая преобразует сообщение в формате неформатированного текста в зашифрованный текст  
ПРАВА ДОСТУПА

**Объектом доступа** (или просто объектом) называют любой элемент операционной

системы, доступ к которому пользователей и других субъектов доступа может быть

произвольно ограничен.

**Методом доступа** к объекту называется операция, определенная для объекта. Тип

операции зависит от объектов.

**Субъектом доступа** называют любую сущность, способную инициировать выполнение

операций над объектами

Для объекта доступа может быть определен **владелец – субъект**

**Правом доступа к** объекту называют право на получение доступа к объекту по

некоторому методу или группе методов.

**Разграничением доступа субъектов** к объектам является совокупность правил,

определяющая для каждой тройки субъект–объект–метод, разрешен ли доступ данного

субъекта к данному объекту по данному методу.

Дальше идёт такая жуть, что выучить это нереально . Страница 405

**32. Механизмы безопасности в операционных системах семейства Windows.**

• Идентификаторы безопасности (SID).

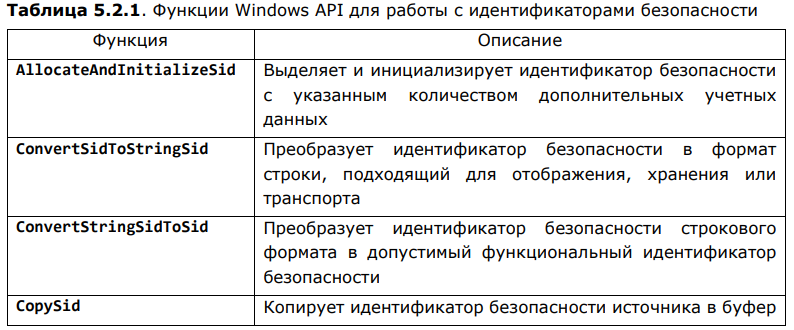
• Маркеры защиты.

• Списки управления доступом (ACL).

• Доменные службы Active Directory.

• Виды групп пользователей.

**Идентификатор безопасности (SID)** — это уникальное значение переменной длины, используемое для идентификации доверенного лица. Каждая учетная запись имеет уникальный идентификатор безопасности, выданный центром сертификации. Каждый раз, когда пользователь входит в систему, система получает идентификатор безопасности для этого пользователя из базы данных и помещает его в маркер доступа для этого пользователя.





**Маркер доступа** представляет собой объект, содержащий информацию о контексте безопасности процесса или потока. Он включает удостоверение и привилегии учетной записи пользователя, связанной с процессом или потоком. После успешной аутентификации пользователя при входе в систему, система создает маркер доступа. Каждый процесс, выполняемый от имени данного пользователя, обладает копией этого маркера доступа. Маркер доступа используется системой для идентификации пользователя при взаимодействии с защищенными объектами или при выполнении системных задач, требующих привилегий.

Маркеры доступа содержат следующие сведения:

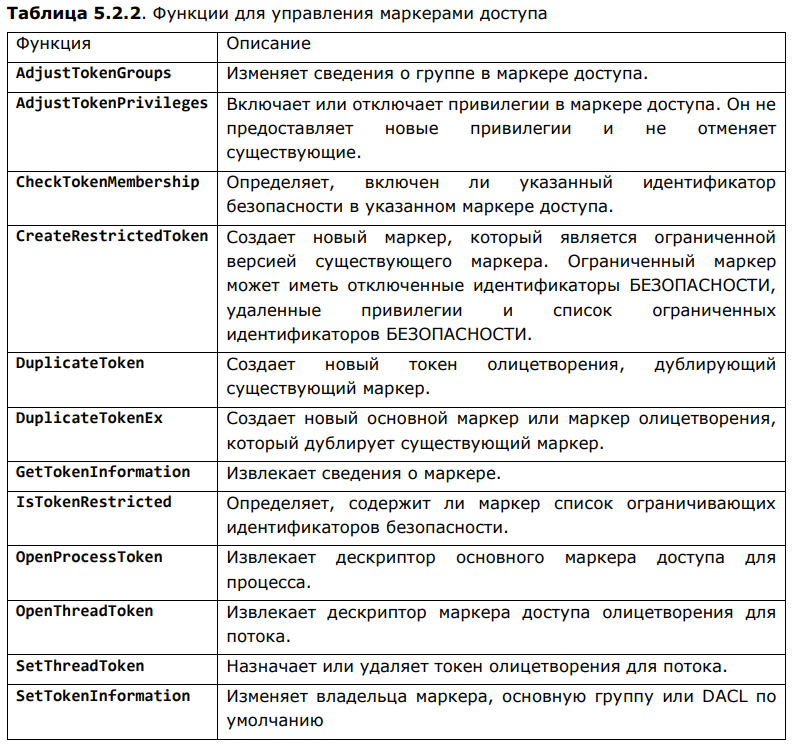
• Идентификатор безопасности (SID) для учетной записи пользователя

• Идентификаторы безопасности для групп, членом которых является пользователь

• Идентификатор безопасности входа, который идентифицирует текущий сеанс входа в систему.

• Список привилегий, которыми пользовались пользователи или группы пользователей.

• Идентификатор безопасности владельца • Идентификатор безопасности для основной группы и тд



**Дескриптор безопасности** содержит сведения о безопасности, связанные с защищаемым объектом.

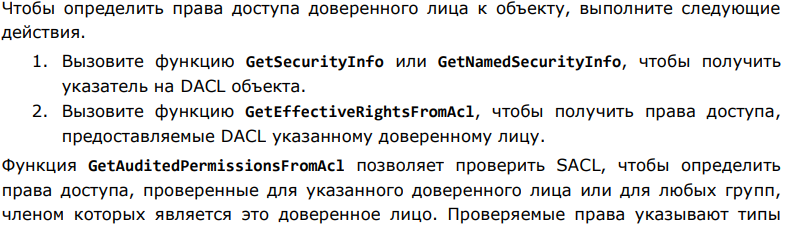
**Списки управления доступом (ACL)** состоят из записей управления доступом (ACE). Каждый ACE в ACL идентифицирует доверенного лица и указывает права доступа, разрешенные, запрещенные или регистрируемые для этого доверенного лица.

Дескриптор безопасности для защищаемого объекта может содержать два типа списков управления доступом: **DACL** и **SACL**

**Список управления доступом (DACL)** на уровне пользователей определяет, кому разрешен или запрещен доступ к объекту. При запросе доступа система проверяет записи ACE в DACL объекта, определяя, следует ли предоставлять доступ. Если DACL отсутствует, полный доступ предоставляется всем пользователям. Если DACL не содержит соответствующих записей ACE, доступ к объекту отклоняется. Система проверяет ACE последовательно, пока не найдет разрешающую запись или отклонит доступ.  
- **\*\*DACL\*\* оп**ределяет доверенные лица, которым разрешен или запрещен доступ к защищаемому объекту.

- **\*\*SACL\*\* позв**оляет администраторам регистрировать попытки доступа и создавать записи в журнале аудита.

**Системный список управления доступом (SACL)** позволяет регистрировать попытки доступа к объекту, создавая записи аудита в журнале событий безопасности. Каждая запись ACE в SACL указывает типы попыток доступа, которые приведут к созданию записи в журнале событий. SACL может фиксировать попытки доступа при успехе и/или при неудаче.



**Active Directory (AD)доменная служба** является службой каталогов, предоставляющей иерархическую структуру для хранения и предоставления сведений об объектах в сети. В контексте AD DS (Active Directory Domain Services), эта служба хранит информацию о пользователях, группах, серверах и других ресурсах в сети. Администраторы и авторизованные пользователи могут получать доступ к этим данным для управления и использования в сети

Active Directory также включает следующие **компоненты**:

• *Схема* — набор правил, который определяет классы объектов и атрибутов, содержащихся в каталоге, ограничения и ограничения экземпляров этих объектов, а также формат их имен.

• *Глобальный каталог*, содержащий сведения о каждом объекте в каталоге. Это позволяет находить сведения о каталоге независимо от того, какой домен в каталоге фактически содержит данные.

• *Механизм запроса и индекса*, чтобы объекты и их свойства могли быть опубликованы и найдены сетевыми пользователями или приложениями.

• *Служба репликации*, которая синхронизирует данные каталога по сети. Все контроллеры домена в домене участвуют в репликации и содержат полную копию всех сведений о каталоге для своего домена. Любые изменения данных каталога реплицируются в домене на все контроллеры домена  
Active Directory имеет два типа групп:

• **Группы безопасности**. Используйте для назначения разрешений общим ресурсам.

• **Группы рассылки**: создание списков рассылки электронной почты.

Каждая группа имеет область, определяющий степень применения группы в дереве домена или лесу. Область группы определяет, где можно предоставить разрешения сети для группы.

Active Directory определяет следующие три области для групп:

• Универсальная(для пользователей и компбютеров из всех доменов в лесу доменов ) • Глобальная(для доступа другого домена ) • Локальная в домене(из любогодомена в лесу доменов,но может быть для доступа к ресурсам в которм она была создана (домене ))управление доступа к ресурсам того домена где была создана \не использовать в других доменах но могут входить пользователи других доменов

**33. Механизмы безопасности в операционных системах семейства Linux.**

В операционной системе GNU/Linux() существуют следующие типы файлов:

• обычные файлы (-) все файлы с данными

• каталог (d) тип файла, данными которого является список имен других файлов и каталогов, вложенных в данный каталог

• символьные ссылки (l) файл, в данных которого содержится адрес другого файла по его имени (а не индексному дескриптору).

• блочные устройства (b)

• символьные устройства (c) файлы устройств предназначены для обращения к аппаратному обеспечению компьютера

• сокеты (s)

• каналы (p)

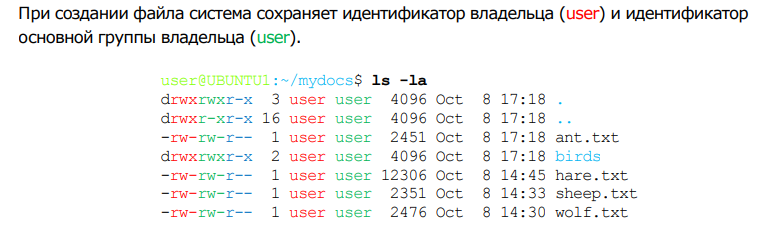
Каждый тип имеет собственное обозначение одним символом.

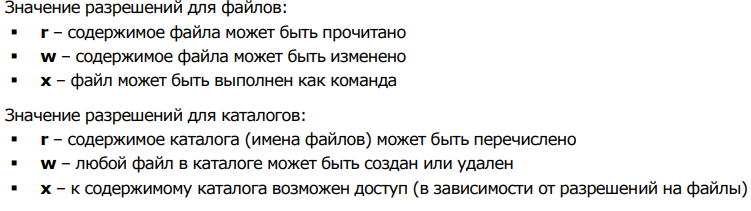
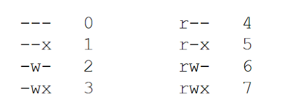
В Linux у каждого файла и каждого каталога есть **два владельца:** *пользователь* и *группа.* владельцы устанавливаются при создании файла или каталога.

Чтобы увидеть текущие назначения владельца, вы можете использовать команду

**ls-l.** Эта команда показывает пользователя и группу-владельца.

Команда **chown** позволяет сменить владельца (команда **chgrp** – только группу-владельца).синий остальные 4 2 1(тире файл d директория)



**Управление разрешениями**

Команда **chmod** позволяет изменять разрешения двумя способами:

* символьным chmod a+x cat.txt
* числовым chmod 750 animals

Некоторые опции:

-R, --recursive изменять файлы и директории рекурсивно

-v, --verbose выводить диагностику о каждом обрабатываемом файле

-c, --changes выводить диагностику только для изменяемых файлов

В операционной системе Linux базовые права для директории равны 0777 (rwxrwxrwx), а для файла 0666 (rw-rw-rw) По умолчанию umask 0002 используется для обычного пользователя. С этой маской права по умолчанию для директории равны 775, а для файла – 664.

**атрибуты файла** — это свойства метаданных, которые описывают поведение файла. Например, атрибут может указывать, сжат ли файл, или указывать, можно ли удалить файл. Вы можете просмотреть атрибуты файла с помощью команды **lsattr**: lsattr todo.txt.

Одно из распространенных применений **chattr** — установка неизменяемого флага для файла или каталога, чтобы пользователи не могли удалить или переименовать файл.

**Управление свойствами файлов**

В Linux каждый файл имеет довольно много свойств, например, права доступа устанавливаются трижды (для владельца, группы и всех прочих), метки времени также бывают трёх разных видов (время создание, доступа и изменения).

У каждого файла доступны следующие метки времени:

• Доступ

• Модифицирован

• Изменён

• Создан

Посмотреть метки времени папки можно также с помощью команды **stat**:

stat /путь/до/папки

С помощью команды **touch** можно изменить три метки времени файла или папки:

• время доступа • время модификации • время изменения статуса