Тема 1.1. Введение в операционные системы

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

##### **Можно ли дать определение операционным системам?**

Так как операционные системы являются очень сложным продуктом и сильно отличаются друг от друга, то строгого и однозначного определения не существует. Проще сказать, чем НЕ является операционная система. Но для определенности, будем придерживаться одного из нижеприведенных определений (на выбор, но заучить).

##### **Определение 1 (предпочтительное)**

Операционная система - это комплект программ, которые служат интерфейсом между модулями вычислительных систем и прикладными программными приложениями, а также управляют компьютерным оборудованием и процессами вычислений, эффективным распределением вычислительных мощностей среди процессов вычислений.

##### **Определение 2**

Операционная система - это комплекс программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

##### **Определение 3**

Операционная система (ОС) - комплекс системных и управляющих программ, предназначенных для наиболее эффективного использования всех ресурсов вычислительной системы (ВС)

##### **Вычислительная система**

Вычислительная система — взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации и удобства работы с ней.

##### **Назначение операционных систем**

ОС управляет всеми устройствами компьютерной системы (процессорами, оперативной памятью, дисками, клавиатурой, монитором, принтерами, сетевыми устройствами и др.) и обеспечивает пользователя удобным интерфейсом для работы с аппаратурой. Две основные функции ОС: 1) предоставлять пользователю некую расширенную виртуальную машину, с которой легче работать (легче программировать), чем непосредственно с аппаратурой реального компьютера или реальной сети; 2) управлять ресурсами вычислительной системы. Поэтому в специальной литературе ОС представляется всегда двояко: как расширенная виртуальная машина и как система управления ресурсами.

##### **Состав операционных систем**

Функции ОС автономного компьютера обычно группируются в соответствии с типами локальных ресурсов, которыми управляет ОС. Такие группы называют подсистемами. Наиболее важные из них: подсистема управления процессами; подсистема управления памятью; подсистема управления файлами; подсистема управления внешними устройствами; подсистема пользовательского интерфейса; подсистема защиты данных и администрирования.

##### **Функции операционных систем**

Некоторые функции ОС: загрузка в ОП программ и их исполнение; инициация программы (передача ей управления); прием и исполнение программных запросов на запуск, приостановку, остановку других программ; организация взаимодействия между задачами; обеспечение работы системы управления файлами и/или систем управления БД; планирование и диспетчеризация задач; управление памятью, организация виртуальной памяти; защита одной программы от влияния другой; обеспечение сохранности данных; аутентификация, авторизация и другие средства обеспечения безопасности.

2. Операционные системы: классификация, основные этапы развития, особенности современного этапа развития.

##### **Классификация ОС по способу организации вычислений**

Системы пакетной обработки - основной задачей является организация наибольшего количества вычислительных процессов за единицу времени. Определенные процессы объединяются в пакет, который затем обрабатывает ОС. Системы разделения времени - создание возможности единовременного взаимодействия с устройством сразу несколькими людьми. В порядке очереди каждый пользователь получает определенный промежуток процессорного времени. Системы реального времени - организация работы каждой задачи за определенный промежуток времени, присущий каждой конкретной задаче.

##### **Классификация ОС количеству единовременно решаемых задач**

Однозадачные; многозадачные.

##### **Классификация ОС возможности работы в компьютерной сети**

Локальные - автономные ОС, которые не позволяют работать с компьютерными сетями; сетевые - ОС с поддержкой компьютерных сетей.

##### **Классификация ОС роли в сетевом взаимодействии**

Серверные - ОС, открывающие доступ к ресурсам сети и осуществляющие управление сетевой инфраструктурой; клиентские - ОС, которые имеют возможность получения доступа к ресурсам сети.

##### **Классификация ОС по типу лицензии**

Открытые – ОС с открытым исходным кодом, который можно изучать и редактировать; проприетарные – ОС, связанные с определенным правообладателем и, как правило, имеющие закрытый исходный код.

##### **Поколения операционных систем**

Этапы развития компьютеров и операционных систем принято классифицировать вместе, так как они взаимосвязаны. Выделяют пять поколений: первое поколение (1945-1955 гг.) - электронные лампы; второе поколение (1955-1965 гг.) - транзисторы; третье поколение (1965-1980 гг.) - интегральные микросхемы; четвертое поколение (1980-2005 гг.) - однокристальные микропроцессоры; пятое поколение (2005 г. - по н.в.) - многоядерные процессоры, смартфоны.

##### **Набор стандартов POSIX**

POSIX (Portable Operating System Interface) - набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-Unix систем.

##### **Эксплуатационные требования к ОС**

К этим требованиям относятся: расширяемость (код операционной системы должен быть написан таким образом, чтобы можно было легко внести дополнения и изменения); переносимость (например, на другой процессор), надежность и отказоустойчивость; совместимость (соответствие стандартам, запуск программ для других ОС); безопасность (защита ресурсов одних пользователей от других); производительность.

##### **Особенности современного этапа развития операционных систем**

На современном этапе развития операционных систем на передний план вышли средства обеспечения безопасности. Это связано с возросшей ценностью информации, обрабатываемой компьютерами, а также с повышенным уровнем угроз, существующих при передаче данных по сетям, особенно по публичным, таким как Интернет. Многие операционные системы обладают сегодня развитыми средствами защиты информации, основанными на шифрации данных, аутентификации и авторизации. Современным операционным системам присуща многоплатформенность, т.е. способность работать на совершенно различных типах компьютеров.

##### **Перспективы развития операционных систем**

Эффективность работы человека становится основным фактором, определяющим эффективность вычислительной системы в целом. Пользовательский интерфейс операционной системы становится все более интеллектуальным, направляя действия человека в типовых ситуациях и принимая за него рутинные решения. Операционные системы будущего должны обеспечить высокий уровень прозрачности сетевых ресурсов, взяв на себя задачу организации распределенных вычислений, превратив сеть в виртуальный компьютер.

Тема 1.2. Архитектура вычислительных систем

3. Компоненты архитектуры вычислительных систем, их назначение и взаимодействие.

Понятие общей шины. Контроллеры. Компоненты компьютера: процессор, память и др.

##### **Шина**

Шина - это несколько проводников, соединяющих несколько устройств. Шины можно разделить на категории в соответствии с выполняемыми функциями. Они могут быть внутренними по отношению к процессору и служить для передачи данных в АЛУ и из АЛУ, а могут быть внешними по отношению к процессору и связывать процессор с памятью или устройствами ввода-вывода. Любая шина состоит из трех частей: шина данных, по которой передается информация; шина адреса, определяющая, кому передаются данные; шина управления, регулирующая процесс обмена информацией. Также может присутствовать шина питания.

##### **Контроллер**

Контроллер - специализированный процессор, управляющий работой внешнего устройства по специальным встроенным программам обмена.

##### **Фон-неймановская архитектура**

ЭВМ, построенная по принципу фон Неймана, содержит следующие основные блоки: 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).

##### **Центральный процессор**

Центральный процессор Центральный процессор выбирает команды из памяти и выполняет их. Обычный цикл работы центрального процессора выглядит так: выборка из памяти первой команды, ее декодирование для определения ее типа и операндов, выполнение этой команды, а затем выборка, декодирование и выполнение последующих команд. Этот цикл повторяется до тех пор, пока не закончится программа. У всех центральных процессоров есть несколько собственных регистров для хранения основных переменных и промежуточных результатов. Среди специальных регистров процессора стоит выделить счетчик команд IP (Instruction Pointer), который содержит адрес ячейки памяти со следующей выбираемой командой, указатель стека SP (Stack Pointer), который ссылается на вершину текущего стека в памяти и слово состояния программы PSW (Program Status Word).

##### **Архитектуры процессоров с различными наборами команд**

Аббревиатура CISC обозначает Complex Instruction Set Computer, а RISC — Reduced Instruction Set Computer. Процессоры CISC имеют большой набор разных команд, включая сложные, а длина команды в байтах для наиболее часто использующихся команд минимальна. Это было вызвано в первую очередь экономией оперативной памяти. Идея RISC заключается в замене сложных инструкций на комбинацию простых. Так не придется заниматься сложной отладкой микрокода. Вместо этого разработчики компилятора будут решать возникающие проблемы. Еще одна основная идея RISC - это конвейеризация. Инструкции разделены на этапы, каждый из которых выполняется примерно одинаковое количество времени.

##### **Конвейер инструкций в RISC-архитектуре**

Если рассмотреть ARM RISC-процессор, то мы обнаружим пятиступенчатый конвейер инструкций. (1 - Fetch) Извлечение инструкции из памяти и увеличение счетчика команд, чтобы извлечь следующую инструкцию в следующем такте. (2 - Decode) Декодирование инструкции — определение, что эта инструкция делает. То есть активация необходимых для выполнения этой инструкции частей микропроцессора. (3 - Execute) Выполнение включает использование арифметико-логического устройства (АЛУ) или совершение сдвиговых операций. (4 - Memory) Доступ к памяти, если необходимо. Это то, что делает инструкция load. (5 - Write Back) Запись результатов в соответствующий регистр.

##### **RISC-V**

RISC-V (RISC-Five), как и RISC и CISC, представляет собой архитектуру набора команд (ISA). Однако это открытая и бесплатная ISA. Это означает, что каждый может внести свой вклад в его развитие, и его использование не будет стоить ни копейки. Это важно, поскольку позволяет мелким производителям устройств создавать оборудование без необходимости платить роялти.

##### **Иерархия памяти**

Используемые компьютером виды памяти перечислены в порядке убывания быстродействия. Верхний уровень состоит из внутренних регистров процессора. Они выполнены по той же технологии, что и сам процессор, и поэтому не уступают ему в быстродействии. Затем следует кэш-память, которая управляется главным образом аппаратурой. Наиболее интенсивно используемые кэш-строки оперативной памяти сохраняются в высокоскоростной кэш-памяти, находящейся внутри процессора или очень близко к нему. Следующей в иерархии идет оперативная память. Это главная рабочая область системы памяти машины. Дополнительно к оперативной памяти многие компьютеры оснащены небольшой по объему неизменяемой памятью с произвольным доступом — постоянным запоминающим устройством (ПЗУ). На некоторых компьютерах в ПЗУ размещается начальный загрузчик, используемый для их запуска. В конце иерархии находится внешняя память: магнитные и твердотельные диски и магнитные ленты.

##### **Магнитные жесткие диски**

Магнитный жесткий диск состоит из одной или нескольких металлических пластин, вращающихся со скоростью 5400, 7200, 10 000 или 15 000 оборотов в минуту. Информация записывается на диск в виде последовательности концентрических окружностей. В каждой заданной позиции привода каждая из головок может считывать кольцеобразный участок, называемый дорожкой. Из совокупности всех дорожек в заданной позиции привода составляется цилиндр. Каждая дорожка поделена на определенное количество секторов, обычно по 512 байт на сектор. На современных дисках внешние цилиндры содержат больше секторов, чем внутренние.

##### **Черепичная магнитная запись**

С целью дальнейшей увеличения плотности записи дорожки стали частично накладывать друг на друга. Такие диски эффективны при последовательной записи данных и редком их чтении (например, системы видеорегистрации). При черепичной магнитной записи (SMR - Shingled Magnetic Recording) на жесткий диск дорожки размещаются друг над другом, подобно черепице на крыше. Это позволяет повысить плотность записи. Увеличивается количество дорожек на дюйм (TPI). При перпендикулярной магнитной записи (PMR), используемой в большинстве современных дисков, данные размещаются на параллельных дорожках. Увеличение TPI путем уменьшения расстояния между дорожками благодаря технологии SMR открывает огромный простор для роста емкости жестких дисков. Конечный продукт физически выглядит и ведет себя как обычный жесткий диск с PMR, но при этом имеет большую емкость. Аббревиатура CMR (Conventional Magnetic Recording, обычная магнитная запись) является антонимом к SMR и по факту эквивалентна PMR.

##### **Твердотельные накопители**

Твердотельные накопители (SSD — Solid State Disks), которые хранят данных во флэш-памяти и не имеют движущихся частей. У них то же назначение, что и у магнитных дисков, и с точки зрения операционной системы выглядят так же. SSD-диски обладают большим быстродействием, чем магнитные, но и стоимость таких дисков выше. Гибридные диски используют обе технологии. SSD-диски могут в одной ячейке хранить несколько бит данных. Чем больше бит данных в одной ячейке, тем выше емкость, ниже цена и меньше количество циклов перезаписи. SLC-ячейки выдерживают до 100000 циклов перезаписи, но хранят в одной ячейке ровно один бит. MLC - 2 бита на ячейку, TLC - 3 бита на ячейку, QLC - 4 бита на ячейку, PLC - 5 битов на ячейку.

##### **PCI Express**

По привычке ее называют шиной, Но по сути это одноранговая сеть, использующая разрядно-последовательные линии и коммутацию пакетов. В шине PCIe соединения между устройствами являются последовательными, то есть имеют разрядность в один бит. Это позволяет значительно увеличить пропускную способность шины. Устройство может иметь до 32 проводных пар, называемых трактами (lanes) или дорожками. Все взаимодействия являются одноранговыми. Когда процессор хочет обратиться к устройству, он отправляет этому устройству пакет и обычно получает ответ. Пакет проходит через корневой комплекс на материнской плате, а затем передается устройству — как правило, через коммутатор (или для устройств PCI - через мост PCI).

##### **Видеокарта**

Видеокарта - устройство, преобразующее цифровую информацию в форму, пригодную для дальнейшего вывода на экран монитора. Обычно видеокарта выполнена в виде печатной платы (плата расширения) и вставляется в слот расширения материнской платы, универсальный либо специализированный (AGP, PCI Express), но может быть реализована и на системной плате. Видеокарты не ограничиваются простым выводом изображения, они имеют встроенный графический процессор, который может производить дополнительную обработку, снимая эту задачу с центрального процессора компьютера. Также имеет место тенденция использовать вычислительные возможности графического процессора для решения неграфических задач (например, добычи криптовалюты).

##### **Цифровые интерфейсы видеокарт**

Разъём HDMI меньше по размеру, чем DVI, а также поддерживает передачу многоканальных цифровых аудиосигналов. При помощи пассивного переходника порт DVI совместим с HDMI, но только в пределах режимов, которым достаточно single-link DVI. Максимальное разрешение DVI 2560x1600. DisplayPort имеет пропускную способность вдвое большую, чем Dual-Link DVI, низкое напряжение питания (3,3 В вместо 5 В, используемых в DVI и HDMI); поддерживает максимальное разрешение 16K (15360x8640). Thunderbolt комбинирует интерфейсы PCI Express (PCIe) и DisplayPort (DP) в одном кабеле.

##### **Direct3D API**

Direct3D — это интерфейс программирования графических приложений (API) для Microsoft Windows. Direct3D, являющийся частью DirectX, используется для рендеринга трехмерной графики в приложениях, где важна производительность, например в играх. Direct3D использует аппаратное ускорение, если оно доступно на видеокарте, что позволяет аппаратно ускорять весь конвейер 3D-рендеринга или даже только частичное ускорение. Текущая версия 12.0.

##### **Задачи операционной системы по управлению и организации работы компьютера**

Самыми важными задачами управления компьютерным оборудованием, осуществляемыми операционной системой, являются следующие: параллельное функционирование модулей ввода, вывода информации и процессора; организация кэширования данных и выполнение согласования скоростей информационного обмена; разбиение модулей и информационных данных среди процессов; организация удобной работы логического интерфейса между модулями и оставшейся частью системы; организация поддержки различных устройств с обеспечением возможности просто их добавить; режим динамической загрузки и выгрузки драйверов; обеспечение поддержки набора файловых систем.

4. Система прерываний.

Аппаратная поддержка в процессорах x86. Различие работы в реальном и защищенном режимах процессора. Векторы прерываний.

##### **Классификация прерываний**

Существует два источника поступления прерываний (interrupt) и три типа особых ситуаций (исключений, exception). Кроме того, различают внутренние (программные, software) и внешние (аппаратные, hardware) источники генерации прерываний и особых ситуаций. В зависимости от источника, прерывания делятся на: аппаратные - возникают как реакция микропроцессора на физический сигнал от некоторого устройства (клавиатура, системные часы, клавиатура, жесткий диск и т.д.), по времени возникновения эти прерывания асинхронны, т.е. происходят в случайные моменты времени; программные - вызываются искусственно с помощью соответствующей команды из программы (int), предназначены для выполнения некоторых действий операционной системы, являются синхронными; исключения - являются реакцией микропроцессора на нестандартную ситуацию, возникшую внутри микропроцессора во время выполнения некоторой команды программы (деление на ноль, прерывание по флагу TF (трассировка)).

##### **Общая классификация прерываний**

Общая классификация прерываний: внешние - вызываются внешними по отношению к микропроцессору событиями (это группа аппаратных прерываний); внутренние - возникают внутри микропроцессора во время вычислительного процесса (по существу это исключительные ситуации и программные прерывания). Внешние прерывания возникают по сигналу какого-нибудь внешнего устройства. Внешние прерывания подразделяются на немаскируемые и маскируемые.

##### **Особые ситуации (исключения)**

Особые ситуации, генерируемые процессором, подразделяются на три типа — ошибки, ловушки и сбои. В зависимости от типа особой ситуации различается реакция процессора на ее возникновение. Ошибка (Fault) - это особая ситуация, которая может быть исправлена обработчиком особой ситуации. При встрече ошибки состояние процессора сохраняется в том виде, каким оно было до начала выполнения команды, инициировавшей генерацию ошибки, а значения CS:EIP, указывающие на эту команду сохраняются в стеке обработчика. Прерванная программа после исправления ошибки может быть продолжена непосредственно с команды, вызвавшей эту ошибку. Ловушка (Trap) - особая ситуация, которая генерируется после выполнения соответствующей команды. В этом случае сохраняемые в стеке значения CS:EIP, указывают на команду, которая будет выполняться вслед за командой, вызвавшей ловушку; например, если ловушка произошла во время команды JMP, то сохраненные значения CS:EIP указывают на команду, являвшуюся целью команды JMP. Сбой (Abort) - это особая ситуация, которая не допускает точную локализацию вызвавшей ее команды и не допускает перезапуска. Сбои используются для сообщений о некоторых ошибках, таких как: технические неисправности и наличие некорректных значений в системных таблицах.

##### **Перечень сокращений для обработки прерываний в RISC-V**

HART (Hardware Thread) - примерно соответствует ядру процессора. MSI (message-signaled interrupts) - прерывания, сигнализируемые сообщениями. IMSIC (Incoming MSI Controller) - контроллер входящего MSI. APLIC (Advanced PLIC, Platform-Level Interrupt Controller) - усовершенствованный контроллер прерываний уровня платформы, действует как традиционный центральный концентратор для прерываний, маршрутизации и определения приоритетов внешних прерываний для каждого порта. IPI (Inter-Processor Interrupt) - межпроцессорное прерывание.

Тема 1.3. Архитектура операционных систем

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

##### **Ресурс**

В первых системах программирования под понятием ресурсы понимали: процессорное время, память, каналы ввода/вывода и периферийные устройства. Впоследствии понятие ресурса стало более универсальным и общим. В настоящее время понятие ресурса превратилось в абстрактную структуру с целым рядом атрибутов, характеризующих способы доступа к этой структуре и ее физическое представление в системе. К ресурсам стали относиться и такие объекты, как сообщения и синхросигналы, которыми обмениваются задачи.

##### **Структура ОС**

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 ядра**

Функции, входящие в состав ядра, можно разделить на два класса. 1 класс. Функции для решения внутрисистемных задач организации вычислительного процесса (переключение контекстов процессов, загрузка/выгрузка страниц, обработка прерываний). Эти функции недоступны для приложений. 2 класс. Функции для поддержки приложений (доступны приложениям). Эти функции создают для приложений так называемую прикладную программную среду и образуют интерфейс прикладного программирования - API. Приложения обращаются к ядру с запросами - системными вызовами. Функции API обслуживают системные вызовы - предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей их физического расположения.

##### **Объекты ядра**

Объекты ядра используются системой и приложениями для управления множеством разных ресурсов: процессами, потоками, файлами и т.д. [9]. Приложение не может напрямую обращаться к объектам ядра читать и изменять их содержимое.

Понятие "объект ядра" имеет разный смысл не только в разных операционных системах, но даже у разных авторов, описывающих одну и ту же операционную систему [10]. К тому же, одни объекты существуют во всех операционных системах (процесс, поток), а другие специфичны для конкретной операционной системы (WindowStation). Здесь будет представлена только общая картина и несколько отрывочных иллюстраций. Многие из объектов ядра будут подробно рассматриваться в соответствующих темах.

**Основные операции**

В состав компонентов объекта типа входит атрибут методы — указатели на внутренние процедуры для выполнения стандартных операций. Методы вызываются диспетчером объектов при создании и уничтожении объекта, открытии и закрытии описателя объекта, изменении параметров защиты.

Система позволяет динамически создавать новые типы объектов. В этом случае предполагается регистрация его методов у диспетчера объектов. Например, метод open вызывается всякий раз, когда создается или открывается объект и создается его новый описатель [10].

Создание новых объектов, или открытие по имени уже существующих, приложение может осуществить при помощи Win32-функций, таких, как CreateFile, CreateSemaphore, OpenSemaphore и т.д. Это библиотечные процедуры, за которыми стоят сервисы Windows и методы объектов. В случае успешного выполнения создается 64-битный описатель в таблице описателей процесса в памяти ядра. На эту таблицу есть ссылка из блока управления процессом EPROCESS.

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

##### **Утилиты**

Утилиты - вспомогательное программное обеспечение — тип системного программного обеспечения, используемого для управления, организации, оптимизации и улучшения функционирования компьютерной системы. Основные типы утилит: Antivirus, File Management System, Disk Management tools, Compression tools, Disk cleanup tool, File Management System, Disk Defragmenter, Backup utility.

##### **Причины выделения машинно-зависимого слоя ОС**

Код операционной системы зависит от аппаратного обеспечения компьютера. Некоторые причины: несовпадение системы команд процессоров; разное количество процессоров (следовательно, разные алгоритмы планирования); особенности реализации (микроконтролеры, количество линий прерываний и т. п.); недоступные средства аппаратной поддержки.

##### **Обрабатывающие системные программы**

Обрабатывающие системные программы отличаются от управляющих программ как по своим функциям, так и по способу их инициирования (запуска). Основные функции обрабатывающих программ: 1) перенос информации. Перенос может выполняться между различными устройствами или в пределах одного устройства. При этом под устройствами понимаются: ОП, устройства ВП, устройства ввода-вывода; 2) преобразование информации. То есть после считывания информации с устройства обрабатывающая программа преобразует эту информацию, а только затем записывает ее на это же или на другое устройство. В зависимости от того, какая из этих двух функций является основной, обрабатывающие системные программы делятся на утилиты и лингвистические процессоры. Основной функцией утилиты является перенос информации, а основная функция лингвистического процессора – перевод описания алгоритма с одного языка на другой. Сущность алгоритма при этом сохраняется, но форма его представления, ориентированная на программиста, преобразуется в форму, ориентированную на ЦП.

##### **Библиотеки**

Библиотеки - это набор функций, которые могут использоваться в различных программах. Библиотеки могут быть статические (библиотека привязывается к определенной программе или софт содержит данную библиотеку в своем теле) и динамическими (библиотеки грузятся в оперативную память и используются).

##### **Программы дополнительных услуг**

В эту категорию входит большое количество разнообразных программ: специальный вариант пользовательского интерфейса, калькулятор, некоторые игры (какие, например, поставляются в составе ОС).

Тема 2.1. Процессы операционных систем

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

##### **Процесс**

Процесс (process) - это некоторая часть (единица) работы, создаваемая операционной системой. Чтобы некоторую часть работы можно было назвать процессом, она должна иметь адресное пространство, назначаемое операционной системой, и идентификатор, или идентификационный номер (id процесса). Процесс должен обладать определенным статусом и иметь свой элемент в таблице процессов. В соответствии со стандартом POSIX он должен содержать один или несколько потоков управления, выполняющихся в рамках его Процесс состоит из множества выполняющихся инструкций, размещенных в адресном пространстве этого процесса. Адресное пространство процесса распределяется между инструкциями, данными, принадлежащими процессу, и стеками, обеспечивающими вызовы функций и хранение локальных переменных.

##### **Системные процессы**

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. Они также выполняют некоторые задачи «по поручению» пользовательских процессов, например, делают запросы на ввод-вывод данных, выделяют память и т.д.

##### **Пользовательские процессы**

Пользовательские процессы выполняют собственный код и иногда обращаются к системным функциям. Выполняя собственный код, пользовательский процесс пребывает в пользовательском режиме (user mode). В пользовательском режиме процесс не может выполнять определенные привилегированные машинные команды.

##### **Основные 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.

##### **Создание процесса в Windows**

В Windows одним вызовом функции Win32 CreateProcess создается процесс, и в него загружается нужная программа. У этого вызова имеется 10 параметров, включая выполняемую программу, параметры командной строки для этой программы, различные параметры безопасности, биты, управляющие наследованием открытых файлов, информацию о приоритетах, спецификацию окна, создаваемого для процесса (если оно используется), и указатель на структуру, в которой вызывающей программе будет возвращена информация о только что созданном процессе.

##### **Создание процесса в Unix**

В UNIX существует только один системный вызов для создания нового процесса — fork. Этот вызов создает точную копию вызывающего процесса. После выполнения системного вызова fork два процесса, родительский и дочерний, имеют единый образ памяти, единые строки описания конфигурации и одни и те же открытые файлы. Обычно после этого дочерний процесс изменяет образ памяти и запускает новую программу, выполняя системный вызов execve или ему подобный. Например, когда пользователь набирает в оболочке команду sort, оболочка создает ответвляющийся дочерний процесс, в котором и выполняется команда sort. Смысл этого двухступенчатого процесса заключается в том, чтобы позволить дочернему процессу управлять его файловыми дескрипторами после разветвления, но перед выполнением execve с целью выполнения перенаправления стандартного ввода, стандартного вывода и стандартного вывода сообщений об ошибках. Для своего завершения процесс может вызвать функцию exit.

##### **Process Control Block**

Информация о процессах хранится в одной структуре данных, которая называется блоком управления процессом PCB (Process Control Block). Блок управления процессом является моделью процесса для операционной системы. Любая операция, производимая операционной системой над процессом, вызывает изменения в PCB. Содержимое PCB между операциями остается постоянным.

##### **Контекст процесса**

Совокупность регистрового, системного и пользовательского контекстов процесса называется контекстом процесса. Регистровым контекстом процесса называется содержимое всех регистров общего назначения процессора (включая значение программного счетчика). Содержимое остальных регистров процессора называется системным контекстом процесса. Информации, получаемой с регистровых и системных контекстов, достаточно для управления работой процесса в операционной системе. С точки зрения пользователя, наибольший интерес вызывает вычислительная деятельность процесса, последовательность преобразования данных и полученные результаты. Пользовательским контекстом называются данные, находящиеся в адресном пространстве процесса.

##### **Стандартные потоки ввода-вывода**

Поток номер 0 (stdin) зарезервирован для чтения команд пользователя или входных данных. Поток номер 1 (stdout) зарезервирован для вывода данных, как правило (хотя и не обязательно) текстовых. Поток номер 2 (stderr) зарезервирован для вывода диагностических и отладочных сообщений в текстовом виде.

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

##### **Организация межпроцессного взаимодействия в ОС.**

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 (mutual exclusion).

Ресурс, который допускает обслуживание только одного пользователя за один раз, называется критическим ресурсом. Если несколько процессов хотят пользоваться критическим ресурсом в режиме разделения времени, им следует синхронизировать свои действия таким образом, чтобы этот ресурс всегда находился в распоряжении не более чем одного их них.

Для организации коммуникации между одновременно работающими процессами применяются средства межпроцессного взаимодействия (Interprocess Communication - IPC).

Выделяются три уровня средств IPC:

• локальный;

• удаленный;

• высокоуровневый.

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

К этому виду IPC принадлежат практически все основные механизмы IPC UNIX, а именно, каналы, разделяемая память и очереди сообщений.

Коммуникационное пространство этих IPC поддерживаются только в пределах локальной системы. Из-за этих ограничений для них могут реализовываться более простые и более быстрые интерфейсы.

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

Сюда относятся удаленные вызовы процедур (Remote Procedure Calls - RPC), сокеты Unix, а также TLI (Transport Layer Interface - интерфейс транспортного уровня) фирмы Sun.

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

Эти пакеты предназначены для переноса уже испытанных протоколов коммуникации приложения на более новую архитектуру.

##### **Сигналы**

Сигнал в операционных системах семейства Unix — асинхронное уведомление процесса о каком-либо событии, один из основных способов взаимодействия между процессами. Когда сигнал послан процессу, операционная система прерывает выполнение процесса, при этом, если процесс установил собственный обработчик сигнала, операционная система запускает этот обработчик, передав ему информацию о сигнале, если процесс не установил обработчик, то выполняется обработчик по умолчанию. Отдельные сигналы подразделяются на три класса: системные сигналы (ошибка аппаратуры, системная ошибка и т.д.); сигналы от устройств; сигналы, определенные пользователем.

##### **Каналы**

Канал (pipe) представляет собой средство связи стандартного вывода одного процесса со стандартным вводом другого. Каналы старейший из инструментов IPC, существующий приблизительно со времени появления самых ранних версий оперативной системы UNIX. Для реализации IPC возможно использование полудуплексных и/или именованных каналов (FIFO).

##### **Тупик**

Классический тупик (deadlock) 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.

##### **Отображение файла в память**

Отображение файла в память (на память) - это такой способ работы с файлами в некоторых операционных системах, при котором всему файлу или некоторой непрерывной части этого файла ставится в соответствие определенный участок памяти (диапазон адресов оперативной памяти). При этом чтение данных из этих адресов фактически приводит к чтению данных из отображенного файла, а запись данных по этим адресам приводит к записи этих данных в файл.

##### **Разделяемая память**

Разделяемая память может быть наилучшим образом описана как отображение участка (сегмента) памяти, которая будет разделена между более чем одним процессом. Это гораздо более быстрая форма IPC, потому что здесь нет никакого посредничества (т.е. каналов и т.п.). Вместо этого, информация отображается непосредственно из сегмента памяти в адресное пространство вызывающего процесса. Сегмент может быть создан одним процессом и впоследствии использован для чтения/записи любым количеством процессов.

##### **Классические проблемы межпроцессного взаимодействия**

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.

В литературе по операционным системам можно встретить множество интересных проблем использования различных методов синхронизации, ставших предметом широких дискуссий и анализа. В данном разделе мы рассмотрим наиболее известные проблемы.

Тема 2.2. Потоки операционных систем

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

##### **Поток выполнения**

Поток выполнения (thread) - 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).

##### **Преимущества потоков**

1. Код программы может выполняться, даже если некоторые действия заблокированы. 2. Легкость и быстрота их создания (по сравнению с процессами). 3. Потоки довольно просто обмениваются данными по сравнению с процессами. 4. Улучшает общую производительность системы при ожиданиях завершения операций ввода-вывода. 5. На многопроцессорных системах могут повысить быстродействие за счет параллельных вычислений.

##### **Недостатки потоков**

1. При программировании приложения с множественными потоками необходимо постоянно думать о потокобезопасности (thread safety). 2. Один неправильно работающий поток может повредить остальные, так как потоки делят общее адресное пространство. 3. Потоки конкурируют друг с другом в адресном пространстве. Стек и thread-local storage, занимая часть виртуального адресного пространства процесса, тем самым делают его недоступным для других потоков.

##### **POSIX Threads**

Это стандарт POSIX-реализации потоков. Стандарт POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995) определяет API для управления потоками, их синхронизации и планирования. Библиотеки, реализующие этот стандарт (и функции этого стандарта), обычно называются pthreads (функции имеют приставку pthread\_). Некоторые из вызовов функций пакета Pthread: pthread\_create - создание нового потока, pthread\_exit - завершение работы вызвавшего потока, pthread\_join - ожидание выхода из указанного потока, pthread\_yield - освобождение центрального процессора, позволяющее выполняться другому потоку, pthread\_attr\_init - создание и инициализация структуры атрибутов потока, pthread\_attr\_destroy - удаление структуры атрибутов потока.

##### **Потоки и процессы в Windows**

В среде Microsoft Windows концепция иная, там процесс - это контейнер для потоков. Процесс-контейнер содержит как минимум один поток. Если потоков в процессе несколько, приложение (процесс) становится многопоточным. Процесс - это исполнение программы. Операционная система использует процессы для разделения исполняемых приложений. Поток - это основная единица, которой операционная система выделяет время процессора.

##### **Классическая модель потоков**

Процесс является способом группировки в единое целое взаимосвязанных ресурсов. У процесса есть адресное пространство, содержащее текст программы и данные, а также другие ресурсы. Эти ресурсы могут включать открытые файлы, необработанные аварийные сигналы, обработчики сигналов, учетную информацию и т. д. Управление этими ресурсами можно значительно облегчить, если собрать их воедино в виде процесса. У потока есть счетчик команд, отслеживающий, какую очередную инструкцию нужно выполнять. У него есть регистры, в которых содержатся текущие рабочие переменные. У него есть стек с протоколом выполнения, содержащим по одному фрейму для каждой вызванной, но еще не возвратившей управление процедуры. Хотя поток может быть выполнен в рамках какого-нибудь процесса, сам поток и его процесс являются разными понятиями и должны рассматриваться по отдельности. Процессы используются для группировки ресурсов в единое образование, а потоки являются сущностью, распределяемой для выполнения на центральном процессоре. Подобно традиционному процессу (то есть процессу только с одним потоком), поток должен быть в одном из следующих состояний: выполняемый, заблокированный, готовый или завершенный. Каждый поток имеет собственный стек.

##### **Многопоточность**

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

##### **Виды взаимодействия параллельных потоков**

Наиболее важные виды: совместное использование разделяемых данных, асинхронное взаимодействие, синхронное взаимодействие. В первом случае необходим взаимно-исключающий доступ к данным - недопустимо, чтобы один поток читал данные, в то время как другой поток их изменял. При асинхронном взаимодействии вводится посредник-буфер между параллельными потоками. Иногда этот способ называют взаимодействием с помощью обмена сообщениями. При синхронном взаимодействии оба взаимодействующих потока подходят к точке синхронизации, обмениваются данными и затем продолжают работать параллельно и независимо. Если один из потоков подошел к точке синхронизации раньше, то он дожидается партнера.

##### **Создание потоков**

В Linux потоки реализованы с помощью системного вызова clone(). Также существуют библиотечные функции, которые служат оберткой для системного вызова clone(). В Windows для создания потоков используется функция CreateThread.

##### **Фоновые потоки в Windows**

В Windows процесс активен до тех пор, пока выполняются его обычные, не фоновые потоки. Это не относится к фоновым потокам. Фоновые потоки немедленно завершаются, как только завершились все активные потоки. Чтобы стать фоновым, поток должен самостоятельно выполнить вызов функции SetThreadPriority с аргументом THREAD\_MODE\_BACKGROUND\_BEGIN. Соответственно, для выхода из фонового режима поток должен выполнить вызов функции SetThreadPriority с аргументом THREAD\_MODE\_BACKGROUND\_END.

**Главный поток процесса**

По умолчанию процесс создается с одним потоком, называемым главным или основным потоком. Потоки в Unix по существу являются дешевой копией процессов и по аналогии с процессами предоставляют механизм для одновременного выполнения нескольких параллельных задач в рамках одного приложения

Техника программирования, позволяющая коду выполняться внутри единого процесса с помощью запуска нескольких потоков называется многопоточностью. Потоки могут выполняться как одновременно, так и нет. Одновременное выполнение потоков одного процесса называется параллелизмом (parallelism). Параллельное выполнение потоков в рамках одного процесса возможно только в многоядерных системах и не является обязательным поведением. В одноядерных системах многопоточность может быть только последовательной. Переводя на программистский язык: многопоточный код не обязан быть по определению быстрым или параллельным, но быть таковым он может.

Все потоки, выполняясь внутри своего процесса, разделяют общую глобальную память — данные и сегменты кучи. Это упрощает обмен данным между потоками процесса. Для этого всего лишь нужно скопировать данные в общие переменные (глобальные или в куче). Если поток изменяет ресурс процесса, это изменение сразу же становится видно другим потокам этого процесса. Тем не менее, каждый поток обладает локальным стеком для хранение локальных данных, которые доступны в рамках только текущего потока.

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. Но при этом следует различать фоновые и обычные, не фоновые потоки. В управляемых потоках (класс **Thread** в .NET) для этого служит свойство **IsBackground**. В чистом Win32 API все потоки создаются как обычные, не фоновые. Чтобы стать фоновым, поток должен самостоятельно выполнить вызов функции **SetThreadPriority** с аргументом THREAD\_MODE\_BACKGROUND\_BEGIN. Соответственно, для выхода из фонового режима поток должен выполнить вызов функции **SetThreadPriority** с аргументом THREAD\_MODE\_BACKGROUND\_END.

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

##### **Диаграммы состояния потоков**

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.

##### **Переключение контекста**

Переключение контекста (context switch) - 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)

В ядре ОС с каждым потоком связаны следующие структуры:

• Общая информация pid, tid, uid, gid, euid, egid и т. д.

• Состояние процесса/потока

• Права доступа

• Используемые потоком ресурсы и блокировки

• Счетчики использования ресурсов (например, таймеры использованного процессорного времени)

• Регионы памяти, выделенные процессу

##### **Планирование задачи**

Планирование задачи - процесс принятия решения, какой задаче передать управление.

Тема 2.3. Диспетчеризация потоков

11. Многозадачность в ОС. Типы многозадачности.

##### **Многозадачность**

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

##### **Типы многозадачности**

Существует два типа многозадачности: процессная многозадачность (основанная на процессах — одновременно выполняющихся программах) и поточная многозадачность (основанная на потоках).

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).

##### **Реентерабельность**

Компьютерная программа в целом или её отдельная процедура называется реентерабельной (reentrant - повторно входимый), если она разработана таким образом, что одна и та же копия инструкций программы в памяти может быть совместно использована несколькими пользователями или процессами. При этом второй пользователь может вызвать реентерабельный код до того, как с ним завершит работу первый пользователь и это как минимум не должно привести к ошибке, а при корректной реализации не должно вызвать потери вычислений. Для обеспечения реентерабельности необходимо выполнение нескольких условий: никакая часть вызываемого кода не должна модифицироваться; вызываемая процедура не должна сохранять информацию между вызовами; если процедура изменяет какие-либо данные, то они должны быть уникальными для каждого пользователя; процедура не должна возвращать указатели на объекты, общие для разных пользователей.

##### **Псевдопараллельная многозадачность**

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

##### **Простое переключение**

Тип многозадачности, при котором операционная система одновременно загружает в память два или более приложений, но процессорное время предоставляется только основному приложению. Для выполнения фонового приложения оно должно быть активизировано.

##### **Совместная (или кооперативная) многозадачность**

Тип многозадачности, при котором следующая задача выполняется только после того, как текущая задача явно объявит себя готовой отдать процессорное время другим задачам. Она использует более передовые методы, чем простое переключение задач. При кооперативной многозадачности приложение может захватить фактически столько процессорного времени, сколько оно считает нужным. Все приложения делят процессорное время, периодически передавая управление следующей задаче.

##### **Вытесняющая (или приоритетная) многозадачность**

Вид многозадачности, в котором операционная система сама передает управление от одной выполняемой программы другой в случае завершения операций ввода-вывода, возникновения событий в аппаратуре компьютера, истечения таймеров и квантов времени, или же поступлений тех или иных сигналов от одной программы к другой. В этом виде многозадачности процессор может быть переключен с исполнения одной программы на исполнение другой без всякого пожелания первой программы и буквально между любыми двумя инструкциями в её коде. Распределение процессорного времени осуществляется планировщиком процессов. К тому же каждой задаче может быть назначен пользователем или самой операционной системой определённый приоритет, что обеспечивает гибкое управление распределением процессорного времени между задачами. Этот вид многозадачности обеспечивает более быстрый отклик на действия пользователя. В основе метода лежат два важнейших и достаточно понятных принципа: квантование времени ЦП и приоритеты потоков.

##### **Квантование**

Квантование означает, что каждому потоку система выделяет определенный интервал времени (квант), в течение которого процессор потенциально может выполнять код этого потока. По завершении выделенного кванта планировщик принудительно переключает процессор на выполнение другого готового потока (если, конечно, такой есть), переводя старый активный поток в состояние готовности. Это гарантирует, что ни один поток не захватит ЦП на непозволительно большое время (как было в более ранних системах с так называемой невытесняющей или кооперативной многозадачностью). Конечно, выделенный квант поток может и не использовать до конца, если в процессе своего выполнения он нормально или аварийно завершится, или потребует наступления некоторого события, или будет прерван системой. Типичный диапазон изменения кванта - от 10 до 50 миллисекунд.

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

##### **Приоритет потока**

Приоритет определяет важность потока и влияет на частоту запуска потока и, возможно, на величину выделяемого кванта. Интуитивно понятно, что потоки могут иметь разную степень важности: системные – более высокую (иначе ОС не сможет решать свои задачи), прикладные – менее высокую. Многие ОС позволяют группировать потоки по их важности, выделяя три группы, или класса: потоки реального времени с максимально высоким уровнем приоритета; системные потоки с меньшим уровнем приоритета; прикладные потоки с самым низким приоритетом. Если приоритет потока может меняться системой, то такие приоритеты называют динамическими, иначе - фиксированными.

##### **Приоритеты потоков в Windows**

Уровни приоритета варьируются от нуля (самый низкий приоритет) до 31 (наивысший приоритет). Система рассматривает все потоки с одинаковым приоритетом как равные. Приоритет каждого потока определяется следующими критериями: класс приоритета процесса, уровень приоритета потока в классе приоритета процесса. Класс и уровень приоритета объединяются для формирования базового приоритета потока. По умолчанию класс приоритета процесса - NORMAL\_PRIORITY\_CLASS, равном 8. Все потоки создаются с уровнем приоритета THREAD\_PRIORITY\_NORMAL.

##### **Динамические приоритеты в Windows**

Каждый поток имеет динамический приоритет. Это приоритет, который использует планировщик, чтобы определить, какой поток следует выполнить. Изначально динамический приоритет потока совпадает с базовым приоритетом. Система может повысить и понизить динамический приоритет, чтобы гарантировать, что она реагирует и что потоки не голодают в течение времени процессора. Система не повышает приоритет потоков с базовым уровнем приоритета от 16 до 31. Динамические повышения приоритета получают только потоки с базовым приоритетом от 0 до 15. Функцию повышения приоритета можно отключить, вызвав функцию SetProcessPriorityBoost или SetThreadPriorityBoost. Чтобы определить, отключена ли эта функция, вызовите функцию GetProcessPriorityBoost или GetThreadPriorityBoost. После повышения динамического приоритета потока планировщик уменьшает этот приоритет на один уровень каждый раз, когда поток завершает квант времени, пока поток не вернется к базовому приоритету. Динамический приоритет потока никогда не меньше базового приоритета.

##### **Приоритеты процессов в Linux**

Существует два вида приоритетов, связанных с каждым процессом - один - это значение "приятности", которое варьируется от -20 (самый высокий приоритет) до 19 (самый низкий приоритет), а другой - приоритет реального времени, варьирующийся от 1 до 99. Когда мы хотим установить приоритет процесса, мы изменяем значение "приятности" процесса. Причем, уменьшать приоритет можно с правами обычного пользователя, но, чтобы его увеличить, нужны права суперпользователя. В результате, планировщик процессов ядра ОС Linux поддерживает приоритеты от 0 (реальное время) до 139 включительно. Приоритеты -20..+19 утилиты или команды nice соответствуют приоритетам 100..139 планировщика процессов.

##### **Менеджер потоков**

В состав ОС включается модуль-планировщик, реализующий выбранные алгоритмы планирования. Поскольку этот модуль представляет собой программный код, то для решения своих задач планировщик должен на некоторое время забирать ЦП. Отсюда следует, что алгоритмы планирования должны быть максимально простыми, иначе возникает опасность, что система будет тратить недопустимо большое время на решение своих внутренних задач, а на выполнение прикладных программ времени не останется. Для реализации приоритетного обслуживания ОС должна создавать и поддерживать набор приоритетных очередей. Для каждого возможного значения приоритета создается своя очередь, в которую потоки (в виде своих дескрипторов) помещаются строго в соответствии с очередностью. Планировщик просматривает эти очереди по порядку следования приоритетов и выбирает для выполнения первый поток в самой приоритетной непустой очереди.

##### **Планирование потоков (диспетчеризация)**

Если в состоянии готовности одновременно находятся два или более процесса или потока, а доступен только один центральный процессор, необходимо выбрать, какой из этих процессов будет выполняться следующим. Та часть операционной системы, на которую возложен этот выбор, называется планировщиком, а алгоритм, который ею используется, называется алгоритмом планирования. Другое название (калька с английского) - диспетчеризация.

##### **Алгоритмы планирования**

Алгоритмы планирования должны обладать следующими общими свойствами: обеспечение максимально возможной загрузки ЦП; обеспечение равномерной загрузки ресурсов вычислительной системы; обеспечение справедливого обслуживания всех процессов и потоков; минимизация времени отклика для интерактивных процессов. По реакции на прерывания по таймеру алгоритмы планирования можно разделить на две категории. Неприоритетный алгоритм планирования выбирает запускаемый процесс, а затем просто дает ему возможность выполняться до тех пор, пока он не заблокируется, или до тех пор, пока он добровольно не освободит центральный процессор. В отличие от этого приоритетный алгоритм планирования предусматривает выбор процесса и предоставление ему возможности работать до истечения некоторого строго определенного периода времени. Если до окончания этого периода он все еще будет работать, планировщик приостанавливает его работу и выбирает для запуска другой процесс.

Тема 2.4. Синхронизация и взаимоблокировка ресурсов

13. Синхронизация и взаимоблокировка ресурсов. Механизмы синхронизации.

Состязательная ситуация. Понятия критического ресурса и области. Механизмы синхронизации: критические секции, мьютексы, семафоры, события, барьеры, атомарные операции

##### **Состязательная ситуация**

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**

Критическая область - часть программы, в которой есть обращение к совместно используемым данным.

##### **Критический ресурс**

Критический ресурс - ресурс, к которому осуществляется одновременный доступ.

##### **Условия для эффективной совместной работы процессов**

1. Два процесса не могут одновременно находиться в своих критических областях. 2. Не должны выстраиваться никакие предположения по поводу скорости или количества центральных процессоров. 3. Никакие процессы, выполняемые за пределами своих критических областей, не могут блокироваться любым другим процессом. 4. Процессы не должны находиться в вечном ожидании входа в свои критические области.

##### **Синхронизация**

Механизм упорядочивания выполнения программных блоков двух или более потоков.

##### **Атомарная операция**

Операция, которая не может быть прервана и выполняется как единое целое.

##### **Особенность синхронизации между процессами**

Синхронизация предполагает наличие общей памяти. В случае потоков проблемы нет, так как потоки разделяют адресное пространство процесса. Но у процессов разобщенные адресные пространства, поэтому нужно найти какое-то решение. 1. Общие структуры хранятся в ядре, для доступа используются системные вызовы. 2. Совместно используемая память. 3 Общий файл. В любом случае несколько процессов, использующих общее адресное пространство, никогда не будут столь же эффективными, как потоки, реализованные на пользовательском уровне, поскольку к управлению процессами неизменно привлекается ядро.

##### **Семафоры**

Целочисленная переменная для подсчета количества активизаций, отложенных на будущее. Значение семафора может быть равно 0, что будет свидетельствовать об отсутствии сохраненных активизаций, или иметь какое-нибудь положительное значение, если ожидается не менее одной активизации.

##### **Мьютексы**

Мьютекс - это совместно используемая переменная, которая может находиться в одном из двух состояний: заблокированном или незаблокированном. Следовательно, для их представления нужен только один бит, но на практике зачастую используется целое число, при этом нуль означает незаблокированное, а все остальные значения — заблокированное состояние. Упрощенная версия семафора.

##### **Критические разделы**

Объект критического раздела обеспечивает синхронизацию, аналогичную той, которая предоставляется объектом мьютекса, за исключением того, что критический раздел может использоваться только потоками одного процесса. Объекты критических разделов нельзя совместно использовать в процессах.

##### **Объект события**

Объект события - это объект синхронизации, состояние которого может быть явно задано с помощью функции SetEvent. Существуют два типа объекта события. Событие сброса вручную. Объект события, состояние которого остается сигнальным до тех пор, пока функция ResetEvent явно не будет сброшена на незначимую. Во время передачи сигнала можно освободить любое количество ожидающих потоков или потоков, которые впоследствии указывают один и тот же объект события в одной из функций ожидания. Событие автоматического сброса. Объект события, состояние которого остается сигнальным до тех пор, пока не будет освобожден один поток ожидания, в этот момент система автоматически устанавливает состояние без знака. Если ожидающих потоков нет, состояние объекта события остается

##### **Заблокированные функции**

Заблокированные функции предоставляют простой механизм для синхронизации доступа к переменной, совместно используемой несколькими потоками. Эти функция является атомарной по отношению к вызовам других взаимосвязанных функций. Они реализуется с помощью встроенной функции компилятора, где это возможно. Примеры функций: InterlockedExchange, InterlockedCompareExchange, InterlockedIncrement, InterlockedDecrement, InterlockedExchangeAdd.

##### **Ресурсы**

Под ресурсом понимается все, что должно предоставляться, использоваться и через некоторое время высвобождаться, поскольку в один и тот же момент времени может использоваться только одним процессом. К выгружаемым относятся такие ресурсы, которые могут быть безболезненно отобраны у процесса, который ими обладает. Примером такого ресурса может послужить память. При недостатке памяти в системе можно забрать память у менее приоритетного процесса и выгрузить ее в файл подкачки. Невыгружаемый ресурс нельзя отобрать у его текущего владельца, не вызвав потенциально сбоя в вычислениях. Обычно это аппаратные устройства, как принтер или накопитель на магнитной ленте. Впрочем, на смартфоне, не поддерживающем свопинг или страничную организацию памяти, простой выгрузкой взаимоблокировки из-за дефицита памяти избежать не удастся.

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

##### **Взаимоблокировка**

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. Поскольку все процессы находятся в состоянии ожидания, ни один из них не станет причиной какого-либо события, которое могло бы возобновить работу другого процесса, принадлежащего к этой группе, и ожидание всех процессов становится бесконечным.

##### **Ресурсная взаимоблокировка**

Ресурсной взаимоблокировкой называется такой вид взаимоблокировки, когда событием, наступления которого ожидает каждый процесс, является высвобождение какого-либо ресурса, которым на данный момент владеет другой участник группы.

##### **Условия возникновения ресурсной взаимоблокировки**

Условия возникновения ресурсной взаимоблокировки Коффман показал, что 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.

##### **Стратегии борьбы с взаимными блокировками**

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.

Тема 2.5. Компьютерное время

15. Компьютерное время. Ожидаемые таймеры.

Часовые пояса. Локальное время и UTC. Эпоха UNIX. Функции POSIX для работы с временем. Ожидаемые таймеры. Протокол NTP.

##### **Определение секунды в Международной системе единиц (СИ)**

Эталон секунды - 9 192 631 770 периодов излучения, соответствующего переходу между двумя сверхтонкими уровнями основного состояния атома цезия-133 при отсутствии возмущения внешними полями.

##### **Международное атомное время**

Международное атомное время (TAI, temps atomique International) - это высокоточный стандарт атомного координатного времени, основанный на воображаемом прохождении собственного времени на геоиде Земли. TAI - это средневзвешенное значение времени, которое показывают более 450 атомных часов в более чем 80 национальных лабораториях по всему миру. Это непрерывная шкала времени, без дополнительных секунд, и это основная реализация земного времени (с фиксированным смещением эпохи).

##### **UTC (всемирное координированное время)**

UTC (всемирное координированное время) - атомная шкала времени, аппроксимирующая UT1. Это международный стандарт, на котором базируется время часовых поясов. В UTC в качестве единицы времени используется секунда СИ, поэтому UTC идёт синхронно с международным атомным временем (TAI). Обычно в сутках UTC 86 400 секунд СИ, но для поддержания расхождения UTC и UT1 не более чем 0,9 с, при необходимости, 30 июня или 31 декабря добавляется (или, теоретически, вычитается) секунда координации.

##### **Социальное время**

Социальное время - это концепция понимания времени в социальных науках и философии. В его основе лежат не циклы движения планет и звёзд, а изменения в обществе, происходящие по воле человека. То есть измеряется оно не единицами продолжительности (минута, час, год), а такими абстрактными мерами, как эпоха, поколение, жизнь. Социальное время отражает не то, сколько длится событие, а то, как ощущается его продолжительность.

##### **Часовые пояса**

Географический часовой пояс - условная полоса на земной поверхности шириной ровно 15° (±7,5° относительно среднего меридиана). Средним меридианом нулевого часового пояса считается гринвичский меридиан. Административный часовой пояс — участок земной поверхности, на котором в соответствии с некоторым законом установлено определённое официальное время.

##### **Unix-время**

Unix-время (также POSIX-время) — система описания моментов во времени, принятая в Unix и других POSIX-совместимых операционных системах. Определяется как количество секунд, прошедших с полуночи (00:00:00 UTC) 1 января 1970 года; этот момент называют эпохой Unix. Время отсчитывается по часовому поясу Гринвичского меридиана без перехода на летнее время (так называемое «Универсальное координированное время» UTC — Coordinated Universal Time). В настоящее время в POSIX-системах тип time\_t эквивалентен типу long. Диапазона значений типа long (32 бита) достаточно для представления дат, лежащих в отрезке длиной примерно 136 лет. Поскольку отрицательные числа используются для представления дат до 1970 года, переполнение текущего представления типа time\_t может произойти в 2038 году, если, конечно, до этого момента представление не будет изменено. Для более удобного представления календарного времени используется тип struct tm, в котором каждый компонент даты и времени хранится в отдельном целочисленном поле.

##### **Основные функции для работы с временем**

Функция time позволяет получить текущее системное календарное время. Функции localtime, gmtime преобразовывают дату из секундного представления time\_t в структурное представление struct tm. Функция mktime конвертирует календарное время местного часового пояса в развёрнутом структурном формате в секундное представление. Функции ctime и asctime конвертируют календарное время в символьную строку. Функция strftime позволяет получать строковое представление времени согласно заданной форматной строке.

##### **Набор часов LINUX**

Ядро таймеров POSIX.1b представляет собой набор часов, которые используются как привязка ко времени. Linux обеспечивает поддержку следующих часов: CLOCK\_REALTIME: общесистемные часы реального времени, видимые для всех процессов, работающих в системе. Часы измеряют количество времени в секундах и наносекундах с начала эпохи. Разрешение часов равно 1/HZ секунд. Таким образом, если HZ равно 100, то разрешение часов составляет 10 мс. Если HZ равно 1000, то разрешение часов составляет 1 мс. Так как это время базируются на времени настенных часов, оно может быть изменено. CLOCK\_MONOTONIC: время непрерывной работы системы, видимое всем процессам в системе. В Linux оно измеряется как количество времени в секундах и наносекундах после загрузки системы. Его разрешение равно 1/HZ с. Это время не может быть изменено каким-либо процессом. CLOCK\_PROCESS\_CPUTIME\_ID: часы, измеряющие время работы процесса. Время текущего процесса, потраченное на выполнение в системе, измеряется в секундах и наносекундах. Разрешение равно 1/HZ. Это время может быть изменено. CLOCK\_THREAD\_CPUTIME\_ID: То же, что и CLOCK\_PROCESS\_CPUTIME\_ID, но для текущего потока. Обычно CLOCK\_REALTIME используется для указания абсолютного времени ожидания. CLOCK\_MONOTONIC используется для относительного времени ожидания и периодических задач. Поскольку время этих часов не может быть изменено, периодическим задачам не нужно беспокоиться о преждевременном или задержанном пробуждении, которое могло бы произойти с CLOCK\_REALTIME. Двое других часов могут использоваться для целей учёта.

##### **Системный таймер**

Системный таймер служит для отслеживания хода времени ядром. Независимо от аппаратной платформы, идея, которая лежит в основе системного таймера, одна и та же - это обеспечение механизма управления прерываниями, которые возникают периодически с постоянной частотой. Для некоторых аппаратных платформ это реализуется с помощью электронных часов, которые генерируют колебания с программируемой частотой. В других аппаратных платформах используется декрементный счетчик (decrementer), куда можно записать некоторое начальное значение, которое будет периодически, с фиксированной частотой, уменьшаться на единицу, пока значение счетчика не станет равным нулю. Когда значение счетчика становится равным нулю, генерируется прерывание. В любом случае эффект получается один и тот же.

##### **Часы реального времени**

Часы реального времени (ЧРВ, RTC - Real Time Clock) — электронная схема, предназначенная для учёта хронометрических данных (текущее время, дата, день недели и др.), представляет собой систему из автономного источника питания и учитывающего устройства. Чаще всего часы реального времени встречаются в вычислительных машинах, хотя на самом деле ЧРВ присутствуют практически во всех электронных устройствах, которые должны хранить время.

##### **Таймер событий высокой точности**

HPET (High Precision Event Timer, таймер событий высокой точности) - тип таймера, используемый в PC-совместимых компьютерах. Совместно разработан Intel и Microsoft, и стал внедряться в чипсеты с 2005 года.

##### **Время Windows**

Время Windows - это количество времени в миллисекундах, прошедшее с момента последнего запуска системы. Функция GetTickCount или GetTickCount64 возвращают текущее время Windows.

##### **Время прерывания**

Время прерывания - это время с момента последнего запуска системы в интервалах по 100 наносекунд. Счетчик времени прерываний начинается с нуля при запуске системы и увеличивается при каждом прерывании часов на длину такта часов. Точная длина тактов часов зависит от базового оборудования и может отличаться в разных системах. В отличие от системного времени, количество прерываний не подлежит корректировке пользователями или службой времени Windows, что делает его лучшим выбором для измерения коротких длительностей.

##### **Таймеры ожидания**

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.

##### **Использование таймеров ожидания**

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. Поток может использовать функцию CancelWaitableTimer для установки таймера в неактивное состояние. Чтобы сбросить таймер, вызовите SetWaitableTimer. Завершив работу с объектом таймера, вызовите CloseHandle, чтобы закрыть дескриптор для объекта таймера.

Тема 3.1. Управление памятью

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

##### **Адресное пространство**

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 использование их адресных пространств).

##### **Виртуальная память**

В основе виртуальной памяти лежит идея, что у каждой программы имеется собственное адресное пространство, которое разбивается на участки, называемые страницами. Каждая страница представляет собой непрерывный диапазон адресов. Эти страницы отображаются на физическую память, но для запуска программы одновременное присутствие в памяти всех страниц необязательно. Когда программа ссылается на часть своего адресного пространства, находящегося в физической памяти, аппаратное обеспечение осуществляет необходимое отображение на лету. Когда программа ссылается на часть своего адресного пространства, которое не находится в физической памяти, операционная система предупреждается о том, что необходимо получить недостающую часть и повторно выполнить потерпевшую неудачу команду.

##### **Диспетчер памяти**

Блок управления памятью (MMU) - это аппаратный компонент компьютера, который обрабатывает все операции с памятью и кэшированием, связанные с процессором. Другими словами, MMU отвечает за все аспекты управления памятью. Обычно он интегрирован в процессор, хотя в некоторых системах он занимает отдельную интегральную схему (ИС).

##### **Страничная организация памяти**

Виртуальное адресное пространство состоит из блоков фиксированного размера, называемых страницами. Соответствующие блоки в физической памяти называются страничными блоками. Страницы и страничные блоки имеют, как правило, одинаковые размеры. Часто их размер составляет 4 Кбайт.

##### **Ошибки отсутствия страниц**

Если программа ссылается на неотображаемые адреса, диспетчер памяти замечает, что страница не отображена, и заставляет центральный процессор передать управление операционной системе. Это системное прерывание называется ошибкой отсутствия страницы (page fault). Операционная система выбирает редко используемый страничный блок и сбрасывает его содержимое на диск (если оно еще не там). Затем она извлекает (также с диска) страницу, на которую была ссылка, и помещает ее в только что освободившийся страничный блок, вносит изменения в таблицы и заново запускает прерванную команду.

##### **Буфер быстрого преобразования адреса**

Translation Lookaside Buffer (TLB) - это тип кэша памяти, в котором хранятся последние трансляции виртуальной памяти в физические адреса для обеспечения более быстрого извлечения. Этот высокоскоростной кэш настроен для отслеживания недавно использованных записей таблицы страниц (PTE). TLB, также известный как кэш трансляции адресов, является частью блока управления памятью процессора (MMU). При использовании TLB нет необходимости размещать PTE в регистрах, что непрактично, или хранить всю таблицу страниц в основной памяти, что требует двух обращений к основной памяти. Вместо этого TLB проверяет, находится ли страница уже в основной памяти. Процессор проверяет TLB на наличие PTE, получает номер кадра и формирует реальный адрес. Если страницы нет в основной памяти, выдается ошибка страницы, и TLB обновляется новым PTE.

##### **Инвертированные таблицы страниц**

В данной конструкции имеется одна запись для каждого страничного блока в реальной памяти, а не одна запись на каждую страницу в виртуальном адресном пространстве. В каждой записи отслеживается, что именно находится в страничном блоке (процесс, виртуальная страница). У них есть один серьезный недостаток: преобразование виртуальных адресов в физические становится намного сложнее. Когда процесс n обращается к виртуальной странице p, аппаратура уже не может найти физическую страницу, используя p в качестве индекса внутри таблицы страниц. Вместо этого она должна провести поиск записи (n, p) по всей инвертированной таблице страниц, причем этот поиск должен быть проведен при каждом обращении к памяти, а не только при ошибках отсутствия страницы. Для ускорения поиска используются TLB и хэш-таблицы.

##### **Рабочий набор процесса**

Рабочий набор процесса - это набор страниц в виртуальном адресном пространстве процесса, которые в настоящее время находятся в физической памяти. Рабочий набор содержит только страничные выделения памяти. В рабочий набор не включаются ресурсы памяти, не допускающие подкачку, такие как расширения адресных окон (AWE) или большие страницы. Если процесс ссылается на страницу памяти, которая в настоящее время не входит в рабочий набор, происходит ошибка страницы. Обработчик ошибки системной страницы пытается устранить ошибку страницы, и в случае успеха страница добавляется в рабочий набор.

##### **NX bit**

Атрибут (бит) NX-Bit (no execute bit в терминологии фирмы AMD) или XD-Bit (execute disable bit в терминологии фирмы Intel) - бит запрета исполнения, добавленный в страницы для реализации возможности предотвращения выполнения данных как кода. Используется для предотвращения уязвимости типа переполнение буфера, позволяющей выполнять произвольный код на атакуемой системе локально или удалённо. Технология требует программной поддержки со стороны ядра операционной системы.

##### **Организация памяти в компьютере**

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

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

##### **Запоминающее устройство**

Запоминающее устройство - носитель информации, предназначенный для записи и хранения данных. В основе работы запоминающего устройства может лежать любой физический эффект, обеспечивающий приведение системы к двум или более устойчивым состояниям.

##### **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. На вершине иерархии находятся регистры процессора. Доступ к ним занимает 0 тактов, но их всего несколько штук. Далее идёт несколько килобайт кэш-памяти первого уровня, доступ к которой занимает примерно 4 такта. Потом идёт пара сотен килобайт более медленной кэш-памяти второго уровня. Потом несколько мегабайт кэш-памяти третьего уровня. 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 лентах.

##### **Оперативная память**

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.

В ПЗУ находятся:

• программа управления работой процессора;

• программа запуска и останова компьютера;

• программы тестирования устройств, проверяющие при каждом включении компьютера правильность работы его блоков;

• 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**

Метод управления памятью процессора, предназначенный для выполнения программ, которым выделяется адресное пространство, превышающее доступный физический объем памяти компьютера.

##### **Свопинг (подкачка)**

Механизм обмена (вытеснения и загрузки) содержимым блоков оперативной физической памяти компьютера с устройством хранения данных с целью расширения адресуемого объема оперативной памяти компьютера. Механизм является аппаратно-программным.

##### **Страничная память**

Реализации виртуальной памяти, при которой физическая память и адресное пространство разбивается на блоки (страницы), а также осуществляется страничный свопинг. Размеры страниц для X86-64: 4K, 2MB, 1GB.

##### **Диспетчер памяти**

Memory Management Unit (MMU) - аппаратное (программируемое) устройство, входящее в состав процессора и предназначенное для трансляции виртуальных адресов оперативной памяти в реальные.

##### **Структура адресного пространства процесса в Windows**

В Windows каждый пользовательский процесс имеет собственное виртуальное адресное пространство. Для компьютеров х86 в 32-разрядных операционных системах виртуальные адреса имеют длину 32 бита, так что каждый процесс имеет 4 Гбайт виртуального адресного пространства, по 2 Гбайта пользователю и ядру. Верхние 2 Гбайт содержат операционную систему (включая код, данные, а также резидентный и нерезидентный пулы). В 64-разрядных версиях Windows нижние 128ТБ адресного пространства выделяются для пользовательского режима, старшие 128ТБ - для режима ядра.

##### **Структура адресного пространства процесса в Linux**

У каждого процесса в системе Linux есть адресное пространство, состоящее из трех логических сегментов: текста, данных и стека. Текстовый сегмент (text segment) содержит машинные команды, образующие исполняемый код программы. Он создается компилятором и ассемблером при трансляции программы (написанной на языке высокого уровня, например, C или C++) в машинный код. Как правило, текстовый сегмент доступен только для чтения. Таким образом, не изменяются ни размеры, ни содержание текстового сегмента. Сегмент данных (data segment) содержит переменные, строки, массивы и другие данные программы. Он состоит из двух частей: инициализированных и неинициализированных данных. По историческим причинам вторая часть называется BSS (Block Started by Symbol). Инициализированная часть сегмента данных содержит переменные и константы компилятора, значения которых должны быть заданы при запуске программы. Все переменные в BSS должны быть инициализированы в нуль после загрузки. Операционная система Linux разрешает сегменту данных расти при выделении памяти и уменьшаться при освобождении памяти. Сегмент стека (stack segment) на большинстве компьютеров начинается около старших адресов виртуального адресного пространства и растет вниз к 0. Например, на 32-битной платформе х86 стек начинается с адреса 0xC0000000, который соответствует предельному виртуальному адресу, видимому процессам пользовательского режима.

##### **Задачи виртуальной памяти**

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.

19. Подкачка. Алгоритмы замещения страниц. Куча (heap). Стек.

##### **Свопинг (подкачка)**

Механизм обмена (вытеснения и загрузки) содержимым блоков оперативной физической памяти компьютера с устройством хранения данных с целью расширения адресуемого объема оперативной памяти компьютера. Механизм является аппаратно-программным.

##### **Алгоритмы замещения страниц**

На лекции были рассмотрены следующие алгоритмы: Оптимальный алгоритм замещения страниц, алгоритм исключения давно использовавшейся страницы, алгоритм «первой пришла, первой и ушла», алгоритм «второй шанс», алгоритм «часы», алгоритм нечастого востребования, алгоритм старения, алгоритм «рабочий набор», алгоритм WSClock. Об этих алгоритмах подробно можно почитать в книге Таненбаума "Современные операционные системы" или в файле Тема\_3\_2\_v13.docx на Diskstation.

##### **Куча**

Куча (heap) — это сегмент памяти, для которого не устанавливается постоянный размер перед компиляцией и который может динамически управляться программистом. То есть, это «свободный пул» памяти, который можно использовать при запуске приложения. Размер кучи приложения определяется физическими ограничениями оперативной памяти (оперативной памяти) и обычно намного больше размера стека. 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().

##### **Управление кучей в Windows**

Каждый процесс имеет кучу по умолчанию, предоставляемую системой. Приложения, которые часто выделяют ресурсы из кучи, могут повысить производительность с помощью частных куч. Частная куча — это блок одной или нескольких страниц в адресном пространстве вызывающего процесса. После создания частной кучи процесс использует такие функции, как HeapAlloc и HeapFree, для управления памятью в этой куче. Функции кучи также можно использовать для управления памятью в куче процесса по умолчанию с помощью дескриптора, возвращаемого функцией GetProcessHeap. Для этой цели новые приложения должны использовать функции кучи вместо глобальных и локальных функций. Нет различий между памятью, выделенной из частной кучи, и памятью, выделенной с помощью других функций выделения памяти. Функция HeapCreate создает частный объект кучи, из которого вызывающий процесс может выделять блоки памяти с помощью функции HeapAlloc. После фиксации страницы не удаляются до завершения процесса или до тех пор, пока куча не будет уничтожена путем вызова функции HeapDestroy. Функция HeapAlloc выделяет указанное количество байтов из частной кучи и возвращает указатель на выделенный блок. Этот указатель можно использовать в функциях HeapFree, HeapReAlloc, HeapSize и HeapValidate.

##### **Стек**

Стек — это сегмент памяти, в котором данные, такие как локальные переменные и вызовы функций, добавляются и/или удаляются по принципу «последним пришел — первым вышел» (LIFO). Вообще говоря, стек — это структура данных, которая хранит значения данных в памяти последовательно. Однако, в отличие от массива, вы получаете доступ (чтение или запись) к данным только на «верхней части» стека. Чтение из стека называется «извлечение» (pop), а запись в стек — «вталкивание» (push).

Тема 3.3. Управление вводом-выводом и файлами

20. Типы устройств ввода/вывода. Обработка внешних прерываний. Синхронный и асинхронный ввод/вывод.

##### **Устройства ввода/вывода**

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, устройства вывода, устройства ввода и вывода. Устройства ввода/вывода (клавиатура, мышь, принтер и т.д.) обычно состоят из механического и электронного компонента, где электронный компонент называется контроллером устройства.

Устройства ввода-вывода можно условно разделить на две категории: блочные устройства и символьные устройства. К блочным относятся такие устройства, которые хранят информацию в блоках фиксированной длины, у каждого из которых есть собственный адрес. Обычно размеры блоков варьируются от 512 до 65 536 байт. Вся передача данных ведется пакетами из одного или нескольких целых (последовательных) блоков. 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.

Другой тип устройств ввода-вывода — 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, сетевые интерфейсы, мыши (в качестве устройства-указателя) и множество других устройств, не похожих на дисковые устройства.

##### **Регистры устройства**

У каждого контроллера для связи с центральным процессором имеется несколько регистров. Путем записи в эти регистры операционная система может давать устройству команды на предоставление данных, принятие данных, включение, выключение или выполнение каких-нибудь других действий. Считывая данные из этих регистров, операционная система может узнать о текущем состоянии устройства, о том, готово ли оно принять новую команду, и т. д. Для доступа к этим регистрам используются два подхода. 1. Каждый регистр получает уникальный номер порта в пространстве ввода-вывода. 2. Регистры отображаются на пространство памяти. Возможен гибридный вариант. Второй подход более удобен для программирования, но при наличии отдельной шины памяти нужно принимать дополнительные меры.

##### **Драйверы устройств**

Драйверы устройств — это программные модули, которые можно подключить к операционной системе для работы с определенным устройством. Операционная система получает помощь от драйверов устройств для работы со всеми устройствами ввода-вывода.

##### **Система ввода-вывода в Linux**

Система ввода-вывода в Linux довольно проста и не отличается от присущих другим UNIX-системам. Как правило, все устройства ввода-вывода выглядят как файлы и доступ к ним осуществляется с помощью тех же системных вызовов read и write, которые используются для доступа ко всем обычным файлам. В некоторых случаях должны быть заданы параметры устройства — это делается при помощи специального системного вызова. В операционной системе Linux все устройства интегрируются в файловую систему в виде так называемых специальных файлов (special files). Каждому устройству ввода-вывода назначается путь (обычно в каталоге /dev). Например, диск может иметь путь /dev/hd1, у принтера может быть путь /dev/lp, а у сети — /dev/net.

##### **Специальные файлы Linux**

Специальные файлы подразделяются на две категории: блочные и символьные. Блочный специальный файл (block special file) состоит из последовательности пронумерованных блоков. Основное свойство блочного специального файла заключается в том, что к каждому его блоку можно адресоваться и получить доступ отдельно. Блочные специальные файлы обычно используются для дисков. Символьные специальные файлы (character special files) обычно используются для устройств ввода или вывода символьного потока. Символьные специальные файлы используются такими устройствами, как клавиатуры, принтеры, сети, мыши, плоттеры и т. д. С каждым специальным файлом связан драйвер устройства, осуществляющий работу с соответствующим устройством. Каждый драйвер разделен на две части, причем обе они являются частью ядра Linux и работают в режиме ядра. Верхняя часть драйвера работает в контексте вызывающей стороны и служит интерфейсом к остальной системе Linux. Нижняя часть работает в контексте ядра и взаимодействует с устройством. Драйверам разрешается делать вызовы процедур ядра для выделения памяти, управления таймером, управления DMA и т. д.

##### **Система ввода-вывода в Windows**

Система ввода-вывода в Windows состоит из служб Plug-and-Play, диспетчера электропитания, менеджера ввода-вывода, а также модели драйвера устройств. Plug-and-Play обнаруживает изменения в конфигурации аппаратного обеспечения, создает (или уничтожает) стеки устройств (для каждого устройства), а также загружает и выгружает драйверы устройств. Диспетчер электропитания настраивает состояние электропитания устройств ввода-вывода, чтобы уменьшить потребление энергии системой, когда устройства не используются. Диспетчер ввода-вывода предоставляет поддержку манипулирования объектами ядра для ввода-вывода, а также операций типа IoCallDrivers и IoCompleteRequest. Однако большая часть работы по поддержке ввода-вывода в Windows реализована в самих драйверах устройств. Чтобы гарантировать, что драйверы устройств хорошо работают с Windows, компания Microsoft описала модель WDM (Windows Driver Model), которой должны соответствовать драйверы устройств. Существует набор разработчика (Windows Driver Kit), который содержит документацию и примеры, помогающие создавать драйверы, соответствующие WDM. Большинство драйверов Windows начинается с копирования подходящего образцового драйвера из WDK и его модификации создателем нового драйвера.

##### **Устройства в Windows**

Устройства в Windows представлены объектами устройств. Объекты устройств используются для представления аппаратных средств (таких, как шины), а также как программные абстракции (наподобие файловых систем, сетевых протоколов и расширений ядра вроде антивирусных драйверов-фильтров). Все они формируют то, что Windows называет стеком устройств.

##### **Обслуживание прерывания в Windows**

Обслуживание прерывания состоит из двух, а иногда и трех этапов. 1. Быстрое сохранение непостоянной информации (например, регистр содержимого) в процедуре прерывания, которая выполняется в IRQL = DIRQL. 2. Обработка сохраненных переменных данных в отложенном вызове процедуры (DPC), который выполняется в IRQL = DISPATCH\_LEVEL. 3. Выполнение дополнительных работ в IRQL = PASSIVE\_LEVEL, если это необходимо.

##### **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.

##### **Буферизация**

Буферизация — метод организации обмена, в частности, ввода и вывода данных в компьютерах и других вычислительных устройствах, который подразумевает использование буфера для временного хранения данных. При вводе данных одни устройства или процессы производят запись данных в буфер, а другие — чтение из него, при выводе — наоборот. Процесс, выполнивший запись в буфер, может немедленно продолжать работу, не ожидая, пока данные будут обработаны другим процессом, которому они предназначены. В свою очередь, процесс, обработавший некоторую порцию данных, может немедленно прочитать из буфера следующую порцию. Таким образом, буферизация позволяет процессам, производящим ввод, вывод и обработку данных, выполняться параллельно, не ожидая, пока другой процесс выполнит свою часть работы. Поэтому буферизация данных широко применяется в многозадачных ОС.

##### **Дисковое планирование**

Способ оптимизации обращения к данным, находящимся на разных участках жесткого диска (обычно магнитного). Известны следующие алгоритмы: случайное планирование, «Первым вошел — первым вышел» (FIFO), с использованием приоритета (PRI), «Последним вошел — первым вышел» (LIFO), стратегия выбора наименьшего времени обслуживания (SSTF), стратегия SCAN, стратегия C-SCAN, стратегия N-step-SCAN, стратегия FSCAN.

21. Файловые системы. Файлы и директории. Управление внешней памятью.

##### **Файловая система**

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.

##### **Файловые системы Linux**

Список файловых систем, которые поддерживаются ядром, находится в файле /proc/filesystems. Виды файловых систем, предлагаемых при установке ОС на базе Linux: Ext, Ext2, Ext3, Ext4, JFS, XFS, Btrfs, Swap.

##### **Файловые системы в Windows**

Типы файловых систем Windows включают: таблица размещения файлов (FAT), FAT32 и расширенная таблица размещения файлов (exFAT); файловая система NT (NTFS); устойчивая файловая система (ReFS).

##### **Файл в системе 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. В именах файлов разрешается использовать все ASCII-символы, кроме символа NUL.

##### **Виртуальная файловая система**

VFS определяет набор основных абстракций файловой системы и разрешенные с этими абстракциями операции. Системные вызовы обращаются к структурам данных VFS, определяют тип файловой системы (к которой принадлежит нужный файл) и при помощи хранящихся в структурах данных VFS указателей на функции запускают соответствующую операцию в указанной файловой системе.

##### **Файл**

Файл - поименованная совокупность данных, обычно размещаемая на внешних запоминающих устройствах.

##### **Дескриптор файла**

Дескриптор файла - это управляющий блок, содержащий информацию, необходимую ОС для различных операций с файлом. Конкретный вид дескриптора файла определяется используемой ОС. Другие названия дескриптора - блок управления файлом, элемент каталога файлов. Типичный дескриптор файла содержит следующие данные: символическое имя файла; тип файла; атрибуты файла; данные о размещении файла во внешней памяти и др.

##### **Индексный дескриптор файла (inode)**

Каждому файлу на диске соответствует один и только один индексный дескриптор файла, который идентифицируется своим порядковым номером - индексом файла (inode). Это означает, что число файлов, которые могут быть созданы в файловой системе, ограничено числом индексных дескрипторов, которое либо явно задается при создании файловой системы, либо вычисляется исходя из физического объема дискового раздела. В частности, он содержит следующую информацию, тип файла и права доступа, идентификатор владельца размер в байтах, отметки времени о доступе, счетчик числа связей.

##### **Кеширование в операционной системе**

Кеширование в операционной системе служит для минимизации количества операций передачи данных. Для достижения данной цели в Linux-системах между дисковыми драйверами и файловой системой имеется кэш. Кэш представляет собой таблицу в ядре, в которой хранятся тысячи недавно использованных блоков. Когда файловой системе требуется блок диска (например, блок i-узла, каталога или данных), то сначала проверяется кэш. Если нужный блок есть в кэше, он берется оттуда, при этом обращения к диску удается избежать (что значительно улучшает производительность системы). Если же блока в кэше страниц нет, то он считывается с диска в кэш, а оттуда копируется туда, куда нужно. Поскольку в кэше страниц есть место только для фиксированного количества блоков, то используется один из описанных ранее алгоритмов замещения страниц.

**Каталоги**

Для удобства файлы могут группироваться в каталоги. Каталоги хранятся на диске в виде файлов, и с ними можно работать практически так же, как с файлами. Каталоги могут содержать подкаталоги, что приводит к иерархической файловой системе.

**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

Тема 4.1. Введение в операционную систему Windows

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. 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. 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. На верхнем уровне операционной системы приложения работают путем взаимодействия с пользователем компьютера, а на нижнем уровне — путем интеграции со службами операционной системы. Вы устанавливаете приложения после установки операционной системы, и для их использования вам необходимо запускать приложения вручную.

##### **Системные службы**

Системные службы включают в себя различные исполнительные службы, которые обеспечивают различные функции в операционной системе. Перечислим некоторые из них. Диспетчер ввода-вывода управляет вводом-выводом. Диспетчер виртуальной памяти управляет виртуализацией памяти в операционной системе. Другие компоненты исполнительной власти контролируют другие аспекты операционной системы. Наборы интерфейсов прикладного программирования (API) позволяют Windows поддерживать различные типы приложений.

##### **Отличия службы от приложения**

Служба – процесс, работающий, даже когда никто не зарегистрирован в системе. Служба использует несколько потоков: один поток принимает команды от операционной системы (Start, Stop, Pause, Resume и т.п.), второй поток является основным рабочим потоком (обычно он создает еще множество потоков для выполнения отдельных задач). Завершение этого потока завершает работу всей службы. Управляется диспетчером управления службами (Service Control Manager, SCM). Физически может находиться в .EXE или .DLL-файле, в одном файле может находиться множество служб. Если служба находится в .DLL-файле, ее запускает svchost.exe.

##### **Реестр Windows**

##### 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.

##### **Серверные версии Windows Server**

Последняя LTSC-версия Windows Server сегодня — это Windows Server 2022. Также поддерживается и Windows Server 2019. Windows Server 2022 обеспечивает расширенную многоуровневую безопасность, гибридные возможности с Azure и гибкую платформу приложений. Он имеет встроенные возможности защищенного ядра, помогающие защитить оборудование, встроенное ПО и возможности ОС Windows Server от расширенных угроз безопасности. Существует три редакции Windows Server 2022: Standard, Datacenter, Datacenter: Azure Edition. В портфолио Microsoft есть еще одна операционная система, основанная на Windows Server, но имеющая другое имя. Это Azure Stack HCI. Она имеет собственную сертификацию оборудования, лицензируется по подписке и может быть установлен только на «голое железо», но не на виртуальную машину.

##### **Клиентские версии Windows**

Различные выпуски Windows удовлетворяют потребности потребителей, от частных лиц до крупных предприятий. Home - Индивидуальное домашнее использование. Pro - Малый и средний бизнес, продвинутые пользователи. Pro для рабочих станций - Пользователи с повышенными требованиями к производительности и хранилищу. Enterprise - Крупные коммерческие организации. Enterprise LTSC - Крупные корпоративные организации с ограничительными требованиями к изменениям. Pro Education - Сравнимо с Pro для сотрудников школы, администраторов, учителей и учащихся. Education - Сравнимо с Enterprise для школьного персонала, администраторов, учителей и учащихся. IoT Core/ Enterprise - Устройства стационарного назначения и бытовые устройства.

##### **Методы инсталляции ОС Windows**

Существует несколько разных вариантов развертывания сервера Windows и настольных операционных систем бизнес-класса. Это могут быть ручная установка, полностью автономные (не обслуживаемые) установки, автономные готовые или дополнительно настроенные установки, а также развертывания заготовленных и, возможно, специально настроенных образов операционной системы.

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 (сценарий регистрации), профили пользователей, сервер сценариев Windows (Windows Scripting Host, WSH) и другие средства. Большинство настроек рабочей среды пользователя можно настроить с помощью групповых политик. Облачные инфраструктуры могут использовать MDM-технологии для настройки клиентских устройств.

##### **Работа с учетными записями пользователей и групп**

Локальные учетные записи пользователей определяются локально на устройстве и могут назначаться только на этом устройстве. Локальные учетные записи пользователей — это субъекты безопасности, которые используются для защиты и управления доступом к ресурсам на устройстве для служб или пользователей. Учетные записи локальных пользователей по умолчанию — это встроенные учетные записи, которые создаются автоматически при установке операционной системы. Учетные записи локальных пользователей по умолчанию не могут быть удалены или удалены и не предоставляют доступ к сетевым ресурсам. Учетные записи локальных пользователей по умолчанию используются для управления доступом к ресурсам локального устройства на основе прав и разрешений, назначенных учетной записи. Учетные записи локальных пользователей по умолчанию и создаваемые локальные учетные записи пользователей находятся в папке Пользователи. Папка Пользователи находится в папке Локальные пользователи и группы в локальной консоли управления Майкрософт (MMC). Управление компьютером — это набор средств администрирования, которые можно использовать для управления локальным или удаленным устройством. Вы также можете управлять локальными пользователями с помощью NET.EXE USER и управлять локальными группами с помощью NET.EXE LOCALGROUP или с помощью различных командлетов PowerShell и других технологий сценариев.

##### **Учетная запись администратора**

Учетная запись локального администратора по умолчанию — это учетная запись пользователя для системного администрирования. У каждого компьютера есть учетная запись администратора (SID S-1-5-домен-500, отображаемое имя — Администратор). Учетная запись администратора — это первая учетная запись, которая создается при установке Windows. 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. Учетную запись администратора по умолчанию нельзя удалить или заблокировать, но ее можно переименовать или отключить.

##### **Учетная запись гостя**

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. По умолчанию учетная запись гостя отключена и имеет пустой пароль. Так как учетная запись гостя может предоставлять анонимный доступ, она считается угрозой безопасности. По этой причине рекомендуется оставить учетную запись гостя отключенной, если только ее использование не требуется. По умолчанию учетная запись "Гость" является единственным членом группы SID S-1-5-32-546 "Гости" по умолчанию, которая позволяет пользователю войти на устройство.

##### **Профили пользователей**

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, перемещаемые, обязательные, временные. Профиль пользователя состоит из следующих элементов: куста реестра HKEY\_CURRENT\_USER и набора папок, хранящегося в файловой системе.

##### **Локальные профили пользователей**

Локальный профиль пользователя создается при первом входе пользователя на компьютер. Профиль хранится на локальном жестком диске компьютера. Изменения, внесенные в профиль локального пользователя, относятся к пользователю и компьютеру, на котором внесены изменения.

##### **Перемещаемые профили пользователей**

Перемещаемый профиль пользователя — это копия локального профиля, которая копируется в общую папку сервера и хранится в ней. Этот профиль загружается на любой компьютер, на который пользователь входит в сеть. Изменения, внесенные в перемещаемый профиль пользователя, синхронизируются с серверной копией профиля при выходе пользователя из системы. Преимущество перемещаемых профилей пользователей заключается в том, что пользователям не нужно создавать профиль на каждом компьютере, который они используют в сети.

##### **Обязательные профили пользователей**

Обязательный профиль пользователя — это тип профиля, который администраторы могут использовать для указания параметров для пользователей. Только системные администраторы могут вносить изменения в обязательные профили пользователей. Изменения, внесенные пользователями в параметры рабочего стола, теряются при выходе пользователя из системы.

##### **Временные профили пользователей**

Временный профиль выдается каждый раз, когда ошибка препятствует загрузке профиля пользователя. Временные профили удаляются в конце каждого сеанса, а изменения, внесенные пользователем в параметры рабочего стола и файлы, теряются при выходе пользователя из системы. Временные профили доступны только на компьютерах под управлением Windows 2000 и более поздних версий.

Тема 4.2. Администрирование в ОС Windows

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

##### **Реестр Windows**

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**

Этот куст содержит информацию о ассоциации файлов и определяет, какое приложение открывается, когда пользователь дважды щелкает определенный тип файла в файловой системе. Например, он определяет, что приложением для файлов .xlsx является Microsoft Excel. Этот куст заполняется на основе настроек, связанных с компьютером и пользователем, которые хранятся в разделах HKEY\_LOCAL\_MACHINE\Software\Classes и HKEY\_CURRENT\_USER\Software\Classes. Обычно вы не будете вносить изменения в этот куст.

##### **HKEY\_CURRENT\_USER**

Этот куст содержит информацию о конфигурации для текущего пользователя, вошедшего в систему. Такие элементы, как цветовая схема Windows пользователя и настройки шрифта, хранятся в соответствующих значениях под этим кустом. При ссылке на этот куст при редактировании реестра этот куст иногда называют HKCU. Этот куст представляет собой ярлык ключа, хранящегося в HKEY\_USERS.

##### **HKEY\_LOCAL\_MACHINE**

Вероятно, это самый важный куст, в который вы, вероятно, будете вносить больше всего изменений. Этот куст, иногда называемый сокращенно HKLM, хранит все параметры конфигурации, связанные с компьютером.

##### **HKEY\_USERS**

Этот куст содержит коллекцию всей информации о конфигурации для всех пользователей, выполнивших вход локально на компьютер, включая текущего пользователя, вошедшего в систему. Фактически, один из ключей под этим кустом — это ключ текущего пользователя, вошедшего в систему, который отображается как куст HKEY\_CURRENT\_USER. Важно знать, что вы, скорее всего, будете вносить прямые изменения в настройки пользователя только для текущего пользователя, вошедшего в систему.

##### **HKEY\_CURRENT\_CONFIG**

Этот куст содержит информацию о текущем профиле оборудования, который локальный компьютер использовал во время запуска системы. Обычно вы не вносите изменения в этот куст.

##### **Ключи, подразделы и значения**

Для поддержания структуры базы данных аналогичные настройки хранятся в папках и подпапках, известных как ключи и подразделы. Это упрощает ссылку на определенное значение реестра. Значения определяют поведение операционной системы и хранятся в разделах и подразделах. Существует много типов значений, в зависимости от типа данных, которые каждое из них хранит. Ниже перечислены наиболее распространенные типы значений реестра. 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, списки управления доступом.

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. Если идентификатор SID используется в качестве уникального идентификатора пользователя или группы, его нельзя использовать повторно для идентификации другого пользователя или группы

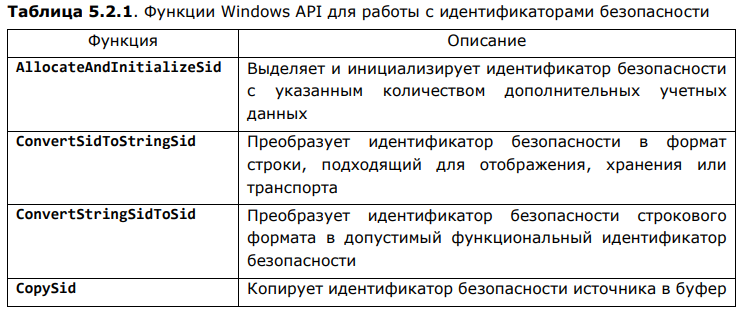
Безопасность Windows использует идентификаторы безопасности в следующих элементах безопасности:

• в дескрипторах безопасности для идентификации владельца объекта и основной группы;

• в записях контроля доступа для определения доверенного лица, для которого разрешен, запрещен или проверен доступ;

• в маркерах доступа — для идентификации пользователя и групп, к которым принадлежит пользователь.





##### **Маркер доступа**

Маркер доступа — это объект, описывающий контекст безопасности процесса или потока.

При каждом входе пользователя система создает маркер доступа для этого пользователя. Маркер доступа содержит идентификатор безопасности пользователя, права пользователя и идентификаторы SID для всех групп, к которому принадлежит пользователь. Этот маркер предоставляет контекст безопасности для любых действий, выполняемых пользователем на этом компьютере.

##### **Списки контроля доступа**

Список управления доступом (ACL) — это список записей управления доступом (ACE). Каждая запись ACE в ACL идентифицирует доверенного лица и указывает разрешенные, запрещенные или проверяемые права доступа для этого доверенного лица. Дескриптор безопасности защищаемого объекта может содержать два типа списков ACL: DACL и SACL.

##### **Список управления дискреционным доступом (DACL)**

Список управления дискреционным доступом (DACL) определяет доверенных лиц, которым разрешен или запрещен доступ к защищаемому объекту. Когда процесс пытается получить доступ к защищаемому объекту, система проверяет записи ACE в списке DACL объекта, чтобы определить, предоставлять ли к нему доступ. Если у объекта нет DACL, система предоставляет полный доступ всем. Если в DACL объекта нет записей ACE, система отклоняет все попытки доступа к объекту, поскольку DACL не разрешает никаких прав доступа. Система последовательно проверяет ACE, пока не найдет один или несколько ACE, которые разрешают все запрошенные права доступа, или пока какое-либо из запрошенных прав доступа не будет отклонено.

##### **Список управления доступом к системе (SACL)**

Список управления доступом к системе (SACL) позволяет администраторам регистрировать попытки доступа к защищенному объекту. Каждый ACE определяет типы попыток доступа со стороны указанного доверенного лица, которые заставляют систему создавать запись в журнале событий безопасности. ACE в SACL может генерировать записи аудита, когда попытка доступа не удалась, когда она увенчалась успехом, или в обоих случаях.

26. Средства автоматической настройки в ОС Windows.

BAT-файлы. Powershell. WMI. ETW. Групповые политики. Административные шаблоны.

##### **Оболочки командной строки Windows**

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. С помощью Windows Script Host можно запускать более сложные скрипты в командной оболочке (команды cscript и vscript). 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. Команды Windows и командлеты PowerShell можно запускать в PowerShell, но командная оболочка может выполнять только команды Windows, а не командлеты PowerShell.

##### **WMI**

Инструментарий управления Windows (WMI) — это инфраструктура для управления данными и операциями в операционных системах Windows. Это расширенная и адаптированная под Windows реализация стандарта WBEM, принятого многими компаниями, в основе которого лежит идея создания универсального интерфейса мониторинга и управления различными системами и компонентами распределённой информационной среды предприятия с использованием объектно-ориентированных идеологий и протоколов HTML и XML. В основе структуры данных в WBEM лежит Common Information Model (CIM), реализующая ООП-подход к представлению компонентов системы. CIM является расширяемой моделью, что позволяет программам, системам и драйверам добавлять в неё свои классы, объекты, методы и свойства. WMI, основанный на CIM, также является открытой унифицированной системой интерфейсов доступа к любым параметрам операционной системы, устройствам и приложениям.

##### **ETW**

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. Архитектура ETW включает в себя 4 элемента: поставщики событий (providers), потребители событий (consumers), контроллеры ETW (controllers), сессии ETW (event tracing sessions). Поставщики событий — это приложения, содержащие инструменты отслеживания событий. Контроллер — это приложение, которое отвечает за функционирование одной или нескольких сессий ETW. Потребители — это приложения, которые получают и обрабатывают события от одного или нескольких сеансов трассировки одновременно. Сеансы отслеживания событий (сессии ETW) записывают события от одного или нескольких провайдеров, которые разрешает контроллер. Сессия также отвечает за управление и очистку буферов.

##### **Групповая политика**

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, в свою очередь, 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; политика контроллеров домена по умолчанию устанавливает базовые параметры безопасности и аудита для всех контроллеров домена в рамках домена.

##### **Применение групповой политики**

Для вступления настроек в силу, объект групповой политики необходимо применить (связать) с одним или несколькими контейнерами Active Directory: сайт, домен или подразделение (OU). Например, можно использовать групповую политику, чтобы потребовать от всех пользователей в определённом домене использовать более сложные пароли или запретить использование съемных носителей на всех компьютерах только в финансовом подразделении данного домена. Объект групповой политики не действует, пока не будет связан с контейнером Active Directory, например, сайтом, доменом или подразделением. Любой объект групповой политики может быть связан с несколькими контейнерами, и, наоборот, с конкретным контейнером может быть связано несколько объектов групповой политики. Кроме того, контейнеры наследуют объекты групповой политики, например, объект групповой политики, связанный с подразделением, применяется ко всем пользователям и компьютерам в его дочерних подразделениях. Аналогичным образом, объект групповой политики, применяемый к OU, применяется не только ко всем пользователям и компьютерам в этом OU, но и наследуется всем пользователям и компьютерам в дочерних OU.

##### **Порядок применения групповой политики**

Настройки различных объектов групповой политики могут перекрываться или конфликтовать. По умолчанию объекты групповой политики обрабатываются в следующем порядке (причем примененные позднее имеют приоритет над примененными ранее): Локальный (индивидуальный компьютер), Сайт, Домен, Организационное подразделение. В эту последовательность можно и нужно вмешиваться, выполнив любое из следующих действий. Изменение последовательности GPO. Объект групповой политики, созданный позднее, обрабатывается последним и имеет наивысший приоритет, перезаписывая настройки в созданных ранее объектах. Это работает в случае возникновения конфликтов. Блокирование наследования. По умолчанию дочерние объекты наследуют все объекты групповой политики от родительского, но вы можете заблокировать это наследование. Принудительное игнорирование связи GPO. По умолчанию параметры родительских политик перезаписываются любыми конфликтующими политиками дочерних объектов. Вы можете переопределить это поведение. Отключение связей GPO. По умолчанию, обработка включена для всех связей GPO. Вы можете предотвратить применение объекта групповой политики для конкретного контейнера, отключив связь с объектом групповой политики этого контейнера.

##### **Административные шаблоны**

Административные шаблоны представляют собой XML-файлы на разных языках, которые определяют основанные на значениях реестра параметры групповых политик, отображаемые в редакторе локальных групповых политик. Существует два вида административных шаблонов: ADMX — не зависящий от языка файл установки, который указывает количество и тип параметров политики, а также расположение по категориям согласно отображению файла в редакторе локальных групповых политик, ADML — файл установки на определенном языке, который предоставляет связанные с языком сведения для ADMX-файла. Этот файл позволяет параметру политики отображаться в редакторе локальных групповых политик на нужном языке. Вы можете добавлять новые языки, добавляя новые ADML-файлы на нужных языках.

Тема 4.3. UNIX-подобные и другие POSIX-совместимые операционные системы

27. Основные понятия системы UNIX. Пользователи системы, атрибуты пользователя. Файловая структура ОС.

##### **Характерные особенности UNIX-систем**

Операционная система UNIX всегда была интерактивной системой, разработанной для одновременной поддержки множества процессов и множества пользователей. Она была разработана программистами и для программистов. От системы требовалось быть простой, элегантной и совместимой. В системе должно быть небольшое количество базовых элементов, которые можно комбинировать, чтобы приспособить их для конкретного приложения. Другие характерные особенности: использование простых текстовых файлов для настройки и управления системой; широкое применение утилит, запускаемых из командной строки; взаимодействие с пользователем посредством виртуального устройства — терминала; представление физических и виртуальных устройств и некоторых средств межпроцессного взаимодействия в виде файлов; использование конвейеров из нескольких программ, каждая из которых выполняет одну задачу.

##### **Основные понятия системы UNIX**

Существует два основных объекта операционной системы UNIX, с которыми приходится работать пользователю – файлы и процессы. Эти объекты сильно связаны друг с другом, и в целом организация работы с ними как раз и определяет архитектуру операционной системы. Все данные пользователя хранятся в файлах; доступ к периферийным устройствам осуществляется посредством чтения и записи специальных файлов; во время выполнения программы, операционная система считывает исполняемый код из файла в память и передает ему управление. С другой стороны, вся функциональность операционная определяется выполнением соответствующих процессов. В частности, обращение к файлам на диске невозможно, если файловая подсистема операционной системы (совокупность процессов, осуществляющих доступ к файлам) не имеет необходимого для этого кода в памяти. В операционной системе содержится большое количество стандартных программ, некоторые из них указаны в стандарте POSIX 1003.2, тогда как другие могут различаться от версии к версии. К этим программам относятся командный процессор (оболочка), компиляторы, редакторы, программы обработки текста и утилиты для работы с файлами. Именно эти программы и запускает пользователь с клавиатуры. Графические интерфейсы пользователя поддерживает оконная система X Windowing System, которую обычно называют Х11 (или просто Х). Она определяет обмен и протоколы отображения для управления окнами на растровых дисплеях UNIX-подобных систем.

##### **Пользователи системы**

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). Каждый пользователь имеет уникальное или регистрационное имя, но система различает пользователей по идентификатору пользователя UID. Идентификаторы также должны быть уникальны. Пользователи являются членами одной или нескольких групп. Информация о пользователях обычно хранится в специальном файле: /etc/passwd. Этот файл доступен только для чтения. Писать в него может только администратор.

##### **Группы**

Группа — список пользователей, имеющих сходные задачи. Принадлежность к группе определяется дополнительными правами, которыми обладают все пользователи группы. Каждая группа имеет уникальное имя, а система различает группы по групповому идентификатору (GID). Информация о группах хранится в специальном файле: /etc/group.

##### **Атрибуты пользователя**

Как правило, 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. passwd-encod - пароль пользователя в закодированном виде. Пароль часто хранят в отдельном файле, а в поле passwd-encod ставится символ х. UID - числовой идентификатор пользователя, является внутренним представлением пользователя в системе. GID - определяет идентификатор первичной группы пользователя. comments - обычно это полное "реальное" имя пользователя. home-dir - домашний каталог пользователя. shell - имя программы, которую UNIX использует в качестве командного интерпретатора.

##### **Файловая структура ОС**

Файлы ядра можно обнаружить в каталоге /boot. В заархивированном виде ядро системы Линукс располагается в vmlinuz, где z - символ, прямо указывающий на то, ядро сжато до минимального размера с целью экономии свободного пространства Файл initrd.img – это первичная файловая система, монтирующая в первую очередь, прежде чем будет осуществлено подключение реальных физических накопители к виртуальной файловой системе VFS. В этом же месте можно обнаружить все другие дополняющие модули ядра, следовательно, данный файл в плане размерности может быть ощутимо больше самого ядра Линукс. В файле system.map не составит труда обнаружить различные функций. Программы устанавливаются в различные директории в файловой системе. Одна из наиболее распространенных директорий для установки программ — это /usr/bin. Здесь обычно размещаются исполняемые файлы программ, доступные для всех пользователей системы. Эта директория имеет преимущество перед другими в пути поиска исполняемых файлов, что позволяет пользователю запускать программы из любой директории. Другая важная директория для установки программ — это /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. Установка программ в эту директорию обеспечивает лучшую изоляцию и восстановление системы после обновления или удаления программ. Наконец, 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. Размещение программ в этой директории позволяет отделить их от программ, поставляемых с операционной системой, и облегчает их управление и обновление.

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

##### **Процессы в Unix и Linux**

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). Несколько независимых процессов могут работать одновременно. Более того, у каждого пользователя может быть одновременно несколько активных процессов, так что в большой системе могут одновременно работать сотни и даже тысячи процессов. Фактически на большинстве однопользовательских рабочих станций работают десятки фоновых процессов, называемых демонами (daemons). Они запускаются при загрузке системы из сценария оболочки. Процессы создаются в операционной системе Linux с помощью системного вызова fork, который 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. Открытые файлы используются родительским и дочерним процессами совместно. Процессы именуются своими PID-идентификаторами. При создании процесса его PID выдается родителю нового процесса. Если дочерний процесс желает узнать свой PID, то он может воспользоваться системным вызовом getpid.

##### **Типы процессов**

Системные процессы являются частью ядра и всегда расположены в оперативной памяти. Демоны — это неинтерактивные процессы, которые запускаются обычным образом — путем загрузки в память соответствующих им программ (исполняемых файлов), и выполняются в фоновом режиме. К прикладным процессам относятся все остальные процессы, выполняющиеся в системе. Как правило, это процессы, порожденные в рамках пользовательского сеанса работы.

##### **Сигналы**

Сигналы — это способ информирования процесса со стороны ядра о происшествии некоторого события. По сути, сигналы являются программными прерываниями. Смысл термина «сигнал» состоит в том, что сколько бы однотипных событий в системе не произошло, по поводу каждой такой группы событий процессу будет подан ровно один сигнал. То есть, сигнал означает, что определяемое им событие произошло, но не несет информации о том, сколько именно произошло однотипных событий. Сигналы могут инициироваться одними процессами по отношению к другим процессам с помощью специального системного вызова kill. 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). Если процесс выбрал перехват посылаемых ему сигналов, он должен указать процедуру обработки сигналов. Когда сигнал прибывает, управление сразу же передается обработчику. Когда процедура обработки сигнала завершает свою работу, управление снова передается в то место, в котором оно находилось, когда пришел сигнал (это аналогично обработке аппаратных прерываний ввода-вывода).

##### **Неименованные каналы**

В системе Linux процессы могут общаться друг с другом с помощью некой формы передачи сообщений. Можно создать канал между двумя процессами, в который один процесс сможет писать поток байтов, а другой процесс сможет его читать. Эти каналы иногда называют трубами (pipes). Синхронизация процессов достигается путем блокирования процесса при попытке прочитать данные из пустого канала. Когда данные появляются в канале, процесс разблокируется. При помощи каналов организуются конвейеры оболочки. Неименованный канал (или программный канал) представляется в виде области памяти на внешнем запоминающем устройстве, управляемой операционной системой, которая осуществляет выделение взаимодействующим процессам частей из этой области памяти для совместной работы, т.е. это область памяти является разделяемым ресурсом.

##### **Именованные каналы**

Файловая система ОС Unix поддерживает специальные файлы, которые называются FIFO-файлами (или именованными каналами). Файлы этого типа очень схожи с обыкновенными файлами (в них можно писать и из них можно читать информацию), за исключением того факта, что они организованы по стратегии FIFO (т.е. невозможны операции, связанные с перемещением файлового указателя). Таким образом, файлы FIFO могут использоваться для организации взаимодействия процессов, при этом в отличие от неименованных каналов эти файлы могут существовать независимо от процессов, взаимодействующих через них. Эти файлы хранятся на внешних запоминающих устройствах, поэтому возможно открыть этот файл, записать в него информацию, а через любой промежуток времени (в течение которого допустимы перезагрузки системы) прочитать записанную информацию.

Тема 4.4. Операционная система ОС Linux

29. Основные принципы функционирования Linux. Основные компоненты Linux. Дистрибутивы Linux. Файловая система Linux.

##### **Основные принципы функционирования Linux**

Организация программно-аппаратных средств во всех UNIX-совместимых системах организована по принципу клиент-сервер. С точки зрения распределения функций, возложенных на систему, все компьютеры в сети работают как один большой компьютер, который может быть легко дополнен аппаратными ресурсами, когда к сети подключается новый компьютер. Каждый пользователь работает с системой через виртуальный терминал, которых может быть до 12-ти в зависимости от версии Linux (на экране обозначается как tty1..tty12). В ОС Linux все файлы организованы в непрерывный поток байтов. Данные, вводимые с клавиатуры, представляют собой входной поток данных, а отображаемые данные – выходной поток. Поскольку процедуры ввода и вывода организованы также, как и файлы, то они могут свободно взаимодействовать с файлами. Чтобы передать данные из одной команды в другую, можно использовать конвейеры. Файл может быть каталогом или исполняемой программой (командой). Команда file помогает определить, для чего используется данный файл (например, текстовый файл или каталог). Файловая система в ОС Linux как и в большинстве других систем имеет иерархическую (древовидную) структуру. Вверху дерева всегда находится корневой каталог ROOT. В этой операционной системе также справедливо понятие текущего каталога. Каждый файл имеет относительное имя пути, которое определяет его принадлежность к какому-либо каталогу, и абсолютное имя пути, которое показывает весь путь файла, начиная от корневого каталога.

##### **Основные компоненты Linux**

Название Linux в первую очередь относится к ядру операционной системы. Вообще говоря, в операционных системах этого семейства можно выделить следующие компоненты: ядро, модули, планировщик, файл подкачки, файловые системы, механизмы защиты, инструменты администрирования, серверные возможности, графический интерфейс пользователя, система управления пакетами и т. д.

##### **Ядро**

В ядре собрана основная функциональность для работы с памятью, управления процессами и т.д. Когда ядру требуется дополнительные функции, он обращается к модулям. В модулях, например, может содержаться код для работы с оборудованием. Файлы ядра Linux находятся в /boot.

##### **Планировщик**

Так как ядро Linux обеспечивает одновременную работу нескольких процессов от нескольких пользователей (с поддержкой нескольких процессоров), операционная система нуждается в средствах управления многопоточностью. Планировщик Linux назначает процессам приоритеты и определяет, какой процесс выполняется на конкретном процессоре (если в системе установлено несколько процессоров). Планировщик можно настроить для работы в системах различного типа. При правильной настройке более важные процессы получают более быструю реакцию процессора. Например, планировщик Linux на настольном компьютере предоставляет больший приоритет задаче перемещения окна и меньший — задаче фонового копирования файлов.

##### **Файл подкачки**

Ядро Linux старается держать работающие в данный момент процессы в оперативной памяти. Простаивающие процессы перемещаются в файл подкачки, представляющий собой выделенную область на жестком диске, которая используется для хранения не перемещающихся в оперативную память данных и процессов. При переполнении оперативной памяти процессы выносятся в файл подкачки. При переполнении файла подкачки (но этого допускать нельзя) новые процессы не запускаются.

##### **Модули**

Ядро Linux поддерживает тысячи аппаратных устройств. При этом за счет включения в работающее ядро только актуальных драйверов размер ядра удается сохранять на приемлемом уровне. Использование загружаемых модулей позволяет добавить в ядро поддержку дополнительных устройств. Модули можно загружать и выгружать по запросу в результате подключения или отключения устройства.

##### **Дистрибутивы Linux**

Дистрибутивы состоят 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:

• в виде бинарных файлов, которые предназначены для непосредственной установки в вашу систему, без какой-либо дополнительной обработки (например, компиляции).

• 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.

##### **Классификация дистрибутивов Linux**

Мы можем распределить дистрибутивы Linux на три группы: Enterprise Grade Linux (Red Hat Enterprise Linux, CentOS, SUSE Linux Enterprise Server, Debian GNU/Linux, Ubuntu LTS); Consumer Grade Linux (Fedora, Ubuntu non-LTS, openSUSE), Experimental and Hacker Linux (Arch, Gentoo). Дистрибутивы Enterprise Grade Linux предназначены для развертывания в крупных организациях с использованием оборудования предприятия. Они обычно включают более старые выпуски ядра и другое программное обеспечение, которое, как известно, работает надежно. Часто дистрибутивы портируют важные обновления, такие как исправления безопасности, на эти стабильные версии. Дистрибутивы Consumer Grade Linux больше ориентированы на малый бизнес или домашних пользователей и любителей. Они подготовлены для использования новейшего оборудования, установленного в системах потребительского уровня. Этим системам потребуются новейшие драйверы, чтобы максимально использовать новое оборудование. Дистрибутивы Experimental and Hacker Linux используют самые современные технологии. Они содержат самые последние версии программного обеспечения, даже если эти версии все еще содержат ошибки и непроверенные функции.

##### **Файловые системы**

Как и UNIX, операционная система Linux изначально создавалась для обеспечения, одновременного многопользовательского доступа. Для защиты пользовательских ресурсов каждому файлу назначаются наборы разрешений на чтение, запись и выполнение, которые определяют права доступа. В стандартной системе Linux пользователь root имеет доступ ко всей системе без ограничений, специальные регистрационные записи могут управлять определенными службами (например, службами Web-сервера Apache), а пользователям могут присваиваться разрешения по отдельности или в составе групп. Последние нововведения, например, Security-Enhanced Linux (SELinux), поддерживают более тонкую настройку и защиту безопасных сред обработки информации.

30. ОС Linux: управление процессами, выполнение задач в фоновом режиме, изменение приоритетов выполняющихся программ.

**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.

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). Это положительные целые числа, которые назначаются в последовательном порядке.

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.

Альтернативные команды – htop и atop.

Команда ps выводит статическую информацию о процессах. Без опций выводит только процессы, относящиеся к текущей оболочке.

Команда поддерживает три стиля параметров, причем они не совпадают.

Например, ps aux не эквивалентна ps –aux, но ps -e эквивалентна ps ax. Иерархию процессов можно увидеть с помощью команды pstree.

Завершить процесс можно с помощью команды kill.

**Выполнение задач в фоновом режиме**

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

В выводе терминала будут отображены порядковый номер задачи (в квадратных скобках) и идентификатор процесса.

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 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

Для завершения фонового процесса применяют команду kill с номером программы.

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 диапазоне 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 Linux поддерживает приоритеты от 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]

У запущенной программы с помощью команды renice можно изменить назначенный приоритет. Предположим, что есть работающая программа yes со значением nice 10. Чтобы изменить его значение, можно использовать команду renice со значением nice и PID процесса. Изменим значение nice на 15:

$ 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.

Тема 5.1. Безопасность и контроль доступа

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

Классификация угроз. Zero trust. Авторизация и аутентификация пользователей. Криптографическая защита данных. Права доступа

##### **Понятие безопасности ОС**

Во многих работах, посвященных безопасности, безопасность информационных систем разбита на три компонента: конфиденциальность, целостность и доступность. Вместе все три компонента часто называют CIA (Confidentiality, Integrity, Availability). Конфиденциальность направлена на сохранение секретности данных. Целостность означает, что пользователи, не обладающие необходимыми правами, не должны иметь возможности изменять какие-либо данные без разрешения их владельцев (в этом контексте изменение данных включает в себя не только внесение в них изменений, но и их удаление или добавление в них ложных данных). Доступность означает, что никто не может нарушить работу системы и вывести ее из строя. Позже были добавлены дополнительные свойства, такие как аутентичность (authenticity), идентифицируемость (accountability), неотвергаемость (nonrepudiability), закрытость (privacy) и др.

##### **Классификация угроз безопасности ОС по цели атаки**

Несанкционированное чтение информации; несанкционированное изменение информации; несанкционированное уничтожение информации; полное или частичное разрушение операционной системы.

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

Использование известных (легальных) каналов получения информации, например, угроза несанкционированного чтения файла, доступ пользователей к которому определен некорректно – разрешен доступ пользователю, которому, согласно политике безопасности, доступ должен быть запрещен; использование скрытых каналов получения информации, например, угроза использования злоумышленником недокументированных возможностей операционной системы; создание новых каналов получения информации с помощью программных закладок.

##### **Классификация угроз по типу используемой злоумышленником уязвимости защиты**

Неадекватная политика безопасности, в том числе и ошибки администратора системы; ошибки и недокументированные возможности программного обеспечения операционной системы, в том числе и так называемые люки – случайно или преднамеренно встроенные в систему «служебные входы», позволяющие обходить систему защиты; ранее внедренная программная закладка.

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

Активное воздействие – несанкционированные действия злоумышленника в системе; пассивное воздействие – несанкционированное наблюдение злоумышленника за процессами, происходящими в системе.

##### **Защищенные операционные системы**

Операционную систему называют защищенной, если она предусматривает средства защиты от основных классов угроз. Защищенная операционная система обязательно должна содержать средства разграничения доступа пользователей к своим ресурсам, а также средства проверки подлинности пользователя, начинающего работу с операционной системой. Кроме того, защищенная операционная система должна содержать средства противодействия случайному или преднамеренному выводу операционной системы из строя. Если операционная система предусматривает защиту не от всех основных классов угроз, а только от некоторых, такую ОС называют частично защищенной.

Существует два основных подхода к созданию защищенных операционных систем – фрагментарный и комплексный

##### **Фрагментарный подход к построению защищенных операционных систем**

При фрагментарном подходе вначале организуется защита от одной угрозы, затем от другой. Примером фрагментарного подхода может служить ситуация, когда за основу берется незащищенная операционная система, на нее устанавливают антивирусный пакет, систему шифрования, систему регистрации действий пользователей и т. д. При применении фрагментарного подхода подсистема защиты операционной системы представляет собой набор разрозненных программных продуктов, как правило, от разных производителей. Эти программные средства работают независимо друг от друга, при этом практически невозможно организовать их тесное взаимодействие. Кроме того, отдельные элементы такой подсистемы защиты могут некорректно работать в присутствии друг друга, что приводит к резкому снижению надежности системы.

##### **Комплексный подход к построению защищенных операционных систем**

При комплексном подходе защитные функции вносятся в операционную систему на этапе проектирования архитектуры операционной системы и являются ее неотъемлемой частью. Отдельные элементы подсистемы защиты, созданной на основе комплексного подхода, тесно взаимодействуют друг с другом при решении различных задач, связанных с организацией защиты информации, поэтому конфликты между ее отдельными компонентами практически невозможны. Подсистема защиты, созданная на основе комплексного подхода, может быть устроена так, что при фатальных сбоях в функционировании ее ключевых элементов она вызывает крах операционной системы, что не позволяет злоумышленнику отключать защитные функции системы. При фрагментарном подходе такая организация подсистемы защиты невозможна. Как правило, подсистему защиты операционной системы, созданную на основе комплексного подхода, проектируют так, чтобы отдельные ее элементы были заменяемы. Соответствующие программные модули могут быть заменены другими модулями.

##### **Административные меры защиты**

Программно-аппаратные средства защиты операционной системы обязательно должны дополняться административными мерами защиты. Без постоянной квалифицированной поддержки со стороны администратора даже надежная программно-аппаратная защита может давать сбои. Перечислим основные административные меры защиты. Постоянный контроль корректности функционирования операционной системы, особенно ее подсистемы защиты. Организация и поддержание адекватной политики безопасности. Осведомление пользователей операционной системы 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.

##### **Архитектура подсистемы защиты операционной системы**

Подсистема защиты ОС выполняет следующие основные функции: идентификация и аутентификация; разграничение доступа; аудит; управление политикой безопасности; криптографические функции; сетевые функции. Подсистема защиты обычно не представляет собой единый программный модуль. Как правило, каждая из перечисленных функций подсистемы защиты решается одним или несколькими программными модулями. Некоторые функции встраиваются непосредственно в ядро ОС.

##### **Идентификация**

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**

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**

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. Авторизация субъекта не относится напрямую к подсистеме защиты операционной системы. В процессе авторизации решаются технические задачи, связанные с организацией начала работы в системе уже идентифицированного и аутентифицированного субъекта доступа.

##### **Криптография**

Криптография – это метод защиты информации путем использования закодированных алгоритмов, хэшей и подписей. Информация может находиться на этапе хранения (например, файл на жестком диске), передачи (например, электронная связь между двумя или несколькими сторонами) или использования (при применении для вычислений). Криптография использует некоторые низкоуровневые криптографические алгоритмы для достижения одной или нескольких из этих целей информационной безопасности. Среди этих инструментов – алгоритмы шифрования, алгоритмы цифровой подписи, алгоритмы хэширования и другие функции.

##### **Алгоритм шифрования**

Алгоритм шифрования – это процедура, которая преобразует сообщение в формате неформатированного текста в зашифрованный текст. Современные алгоритмы используют сложные математические вычисления и один или несколько ключей шифрования. Благодаря этому можно относительно легко зашифровать сообщение, но практически невозможно расшифровать его, не зная ключей. В зависимости от того, как действуют ключи, технологии шифрования делятся на две категории: симметричные и асимметричные.

##### **Криптография с симметричным ключом**

Алгоритмы шифрования с симметричным ключом используют одни и те же криптографические ключи для шифрования простого текста и расшифровки зашифрованного. При использовании симметричного шифрования все получатели сообщения должны иметь доступ к общему ключу.

##### **Асимметричная криптография (с открытым ключом)**

Схема шифрования называется асимметричной, если в ней один ключ (открытый) используется для шифрования данных, а другой, но математически связанный (частный) – для их расшифровки. Необходимо, чтобы было невозможно вычислить частный ключ, если известен только открытый. Поэтому общий открытый ключ можно передавать, а частный – держать в тайне и в безопасности. Эти ключи называются парой ключей.

##### **Гибридное шифрование**

Сочетание криптографии с открытым ключом для обмена ключей и симметричного шифрования для пакетного шифрования данных называется гибридным шифрованием. В гибридном шифровании используются уникальные свойства криптографии с открытым ключом для обмена секретной информацией по недоверенному каналу с эффективностью симметричного шифрования. Оно представляет собой практически применимое сквозное решение для обеспечения конфиденциальности данных. Гибридное шифрование широко используется в протоколах передачи данных для Интернета, таких как протокол TLS (безопасность транспортного уровня). Когда вы подключаетесь к веб-сайту, который использует HTTPS (безопасный HTTP с TLS), браузер согласовывает криптографические алгоритмы, защищающие соединение. Это алгоритмы обмена ключами, симметричного шифрования и цифровой подписи.

##### **Криптографическая хэш-функция**

Криптографическая хэш-функция – это инструмент для преобразования произвольных данных в «отпечаток» фиксированной длины. Хэш-функции создаются таким образом, чтобы было сложно найти два различных набора входных данных, дающих один и тот же отпечаток, и чтобы было сложно найти сообщение, отпечаток которого совпадает с фиксированным значением. В отличие от схем шифрования, схем подписей и MAC, хэш-функции не имеют ключа. Кто угодно может вычислить хэш для данного входного значения, и хэш-функция всегда будет генерировать одно и то же самое выходное значение для одного и того же входного.

##### **Права доступа**

Домен (domain) представляет собой множество пар (объект, права доступа). Каждая пара определяет объект и некоторое подмножество операций, которые могут быть выполнены в отношении этого объекта. Права доступа (rights) означают в данном контексте разрешение на выполнение той или иной операции. Зачастую домен соотносится с отдельным пользователем, сообщая о том, что может, а что не может сделать этот пользователь, но он может также иметь и более общий характер, распространяясь не только на отдельного пользователя. К примеру, сотрудники одной группы программистов, работающие над одним и тем же проектом, могут целиком принадлежать к одному и тому же домену и иметь доступ к файлам проекта.

Тема 5.2. Механизмы безопасности в операционных системах семейства Windows

32. Механизмы безопасности в операционных системах семейства Windows.

Идентификаторы безопасности (SID). Маркеры защиты. Списки управления доступом (ACL). Доменные службы Active Directory. Виды групп пользователей.

##### **Идентификаторы безопасности (SID)**

Идентификатор безопасности используется для уникальной идентификации субъекта безопасности или группы безопасности. Субъекты безопасности могут представлять любую сущность, которая может быть проверена операционной системой, например учетной записью пользователя, учетной записью компьютера или потоком или процессом, выполняющимся в контексте безопасности учетной записи пользователя или компьютера. Каждая учетная запись или группа или каждый процесс, выполняемый в контексте безопасности учетной записи, имеет уникальный идентификатор безопасности, выданный центром, например контроллером домена Windows. Идентификатор безопасности хранится в базе данных безопасности. Система создает идентификатор безопасности, определяющий определенную учетную запись или группу во время создания учетной записи или группы. Если идентификатор безопасности используется в качестве уникального идентификатора для пользователя или группы, он никогда не может использоваться повторно для идентификации другого пользователя или группы.

##### **Маркер доступа**

При каждом входе пользователя система создает маркер доступа для этого пользователя. Маркер доступа содержит идентификатор безопасности пользователя, права пользователя и идентификаторы SID для всех групп, к которому принадлежит пользователь. Этот маркер предоставляет контекст безопасности для любых действий, выполняемых пользователем на этом компьютере.

##### **Списки контроля доступа**

Список управления доступом (ACL) — это список записей управления доступом (ACE). Каждая запись ACE в ACL идентифицирует доверенного лица и указывает разрешенные, запрещенные или проверяемые права доступа для этого доверенного лица. Дескриптор безопасности защищаемого объекта может содержать два типа списков ACL: DACL и SACL.

##### **Список управления дискреционным доступом (DACL)**

Список управления дискреционным доступом (DACL) определяет доверенных лиц, которым разрешен или запрещен доступ к защищаемому объекту. Когда процесс пытается получить доступ к защищаемому объекту, система проверяет записи ACE в списке DACL объекта, чтобы определить, предоставлять ли к нему доступ. Если у объекта нет DACL, система предоставляет полный доступ всем. Если в DACL объекта нет записей ACE, система отклоняет все попытки доступа к объекту, поскольку DACL не разрешает никаких прав доступа. Система последовательно проверяет ACE, пока не найдет один или несколько ACE, которые разрешают все запрошенные права доступа, или пока какое-либо из запрошенных прав доступа не будет отклонено.

##### **Список управления доступом к системе (SACL)**

Список управления доступом к системе (SACL) позволяет администраторам регистрировать попытки доступа к защищенному объекту. Каждый ACE определяет типы попыток доступа со стороны указанного доверенного лица, которые заставляют систему создавать запись в журнале событий безопасности. ACE в SACL может генерировать записи аудита, когда попытка доступа не удалась, когда она увенчалась успехом, или в обоих случаях.

##### **Логическая структура Active Directory**

Логическая структура Active Directory представляет собой иерархическую структуру доменов Active Directory и подразделений в лесу. Взаимосвязь компонентов логической структуры контролирует доступ к хранимым данным, а также то, как информация реплицируется между различными контроллерами домена в лесу.

##### **Лес**

Лес — это высший уровень иерархии логических структур. Лес Active Directory представляет собой отдельный автономный каталог. Лес — это граница безопасности. Это означает, что администраторы леса имеют полный контроль над всем доступом к информации, хранящейся внутри леса, и к контроллерам домена, которые используются для реализации леса.

##### **Домен**

Домены разделяют информацию, хранящуюся внутри каталога, на более мелкие части, чтобы информацию было легче хранить на различных контроллерах домена, а администраторы имели большую степень контроля над репликацией. Данные, хранящиеся в каталоге, реплицируются по всему лесу от одного контроллера домена к другому. Некоторые данные, относящиеся ко всему лесу, реплицируются на все контроллеры домена. Другие данные, относящиеся только к определенному домену, реплицируются только на контроллеры домена в этом конкретном домене. Хороший дизайн домена позволяет реализовать эффективную топологию репликации. Это важно, поскольку позволяет администраторам управлять потоком данных по сети, то есть контролировать, сколько данных реплицируется и где происходит этот трафик репликации.

##### **Подразделения (OU)**

Подразделения предоставляют администраторам возможность группировать ресурсы, такие как учетные записи пользователей или учетные записи компьютеров, чтобы ресурсами можно было управлять как одним блоком. Это значительно упрощает применение групповой политики к нескольким компьютерам или управление доступом многих пользователей к одному ресурсу. Подразделения также упрощают делегирование контроля над ресурсами различным администраторам.

##### **Схема Active Directory**

Все, что хранится в Active Directory, хранится в объекте. В схеме хранится определение каждого типа объекта. Сами определения состоят из двух типов объектов: объектов классов и объектов атрибутов. Классы определяют группы атрибутов, которые используются для описания общих объектов. Определения новых объектов создаются путем объединения различных объектов классов и объектов атрибутов для создания новых комбинаций, содержащих необходимые атрибуты для удовлетворения требований к хранению нового типа объекта.

##### **Группы безопасности**

Группа безопасности — это совокупность учетных записей пользователей, учетных записей компьютеров и других групп учетных записей, которыми с точки зрения безопасности можно управлять как единым целым. В операционных системах Windows существует несколько встроенных групп безопасности, которым предварительно настроены соответствующие права и разрешения для выполнения определенных задач. Кроме того, вы можете создавать группу безопасности для каждой уникальной комбинации требований безопасности, которая применяется к нескольким пользователям в вашей организации. Группы могут быть основаны на Active Directory или локальны для определенного компьютера: Группы безопасности Active Directory используются для управления правами и разрешениями на ресурсы домена. Локальные группы существуют в базе данных SAM на локальных компьютерах (на всех компьютерах под управлением Windows), за исключением контроллеров домена. Вы используете локальные группы для управления правами и разрешениями только для ресурсов на локальном компьютере.

##### **Группы безопасности Active Directory**

Active Directory имеет два типа групп: группы безопасности (для назначения разрешений для общих ресурсов) и группы рассылки (для создания списков рассылки электронной почты). У каждой группы есть область, определяющая степень применения группы в дереве доменов или лесу. Область действия группы определяет, где в сети могут быть предоставлены разрешения для группы. Active Directory определяет следующие три области групп: универсальная, глобальная, локальная в домене.

Тема 5.3. Механизмы безопасности в операционных системах семейства Linux

33. Механизмы безопасности в операционных системах семейства Linux.

Типы файлов в ОС Linux. Владельцы файлов. Управление правами доступа в файловой системе. Атрибуты файлов. Управление свойствами файлов.

##### **Типы файлов в ОС Linux**

В GNU/Linux, как и других Unix-подобных операционных системах, понятие типа файла не связано с расширением файла (несколькими буквами после точки в конце имени), как это обстоит в Windows. Поэтому тип файла в Linux – это скорее тип объекта, но не тип данных, как в Windows. В операционной системе GNU/Linux существуют следующие типы файлов: обычные файлы, каталог, символьные ссылки, блочные устройства, символьные устройства, сокеты, каналы. Каждый тип имеет собственное обозначение одним символом.

##### **Обычные файлы**

Сюда относятся все файлы с данными, играющими роль ценной информации сами по себе. Linux не различает текстовые файлы, исполняемые или картинки. В любом случае это будет обычный (regular) файл. Все они обозначаются знаком "-". Остальные типы файлов считаются специальными (special).

##### **Каталоги**

В Linux каталог представляет собой такой тип файла, данными которого является список имен других файлов и каталогов, вложенных в данный каталог. Напрямую, то есть через какой-либо редактор, пользователь не может редактировать данные файла-каталога. Редактированием занимается ядро операционной системы, получая, в том числе от пользователя, команды создания файла, удаления и др. В файле каталога осуществляется связь между именами файлов (словесного обозначения для людей) и их индексными дескрипторами (истинным именем-числом, которым оперирует ОС). В Unix-подобных системах один и тот же файл может существовать под разными именами и/или в разных каталогах: все имена будут связаны с одним и тем же индексным дескриптором (механизм жестких ссылок). Также следует, что файлы всегда должны содержаться в каталогах, иначе станут недоступны, так как нигде не будет содержаться записи о них.

##### **Символьные ссылки**

Символьная ссылка – это файл, в данных которого содержится адрес другого файла по его имени (а не индексному дескриптору). Выполнение символьной ссылки приводит к открытию файла, на который она указывает. Это аналог ярлыков в операционной системе Windows. Если удалить исходный файл, то символьная ссылка продолжит существовать. Она по-прежнему будет указывать на файл, которого уже нет. Символьные ссылки не содержат атрибутов файлов, на которые они указывают. У них есть собственные атрибуты (свое время создания, размер, права доступа).

##### **Команда file**

Линукс не делает предположение о типе данных в обычном файле, но есть специальная утилита, которая выполняет эту задачу, – программа file. Для этого она анализирует начало содержимого файла и находит в нем специальные "сигналы", характерные для определенного типа – бинарного файла, текстового, изображения и др.

##### **Владельцы файлов**

В Linux у каждого файла и каждого каталога есть два владельца: пользователь и группа. Эти владельцы устанавливаются при создании файла или каталога. Пользователь, который создаёт файл становится владельцем этого файла, а первичная группа, в которую входит этот же пользователь, так же становится владельцем этого файла.

##### **Разрешения на доступ к файлу**

По сути, разрешения задаются в восьмеричной системе. В числовом представлении они записываются как трех- (или иногда четырех-)значное число. В символьном представлении каждая цифра записывается как три символа: старший бит r, средний бит w, младший бит x, их численные значения соответственно 4, 2, 1. Три крайних правых цифры в числовом представлении относятся к следующим категориям пользователей: первая относится к пользователю-владельцу, вторая – к группе-владельцу, третья – ко всем остальным пользователям. Применяется наиболее точное разрешение. В этом же порядке следуют группы разрешений в символьном представлении, отсутствующие разрешения обозначаются дефисом. Значение разрешений для файлов: r – содержимое файла может быть прочитано, w – содержимое файла может быть изменено, x – файл может быть выполнен как команда. Значение разрешений для каталогов: r – содержимое каталога (имена файлов) может быть перечислено, w – любой файл в каталоге может быть создан или удален, x – к содержимому каталога возможен доступ (в зависимости от разрешений на файлы).

##### **Управление разрешениями**

Команда chmod позволяет изменять разрешения двумя способами: символьным и числовым.

##### **Специальные разрешения**

Эффект для файлов: u+s (suid) – файл выполняется от имени владельца файла, а не фактического пользователя; g+s (sgid) – файл выполняется от имени группы-владельца файла; o+t (sticky) – нет эффекта. Эффект для каталогов: u+s (suid) – нет эффекта; g+s (sgid) – создаваемые в каталоге файлы в качестве группы-владельца получат группу-владельца каталога; o+t (sticky) – пользователи с разрешением write могут удалять только свои собственные файлы.

##### **Атрибуты файлов**

В Linux атрибуты файла — это свойства метаданных, которые описывают поведение файла. Например, атрибут может указывать, сжат ли файл, или указывать, можно ли удалить файл. Некоторые атрибуты, такие как неизменяемость, могут быть установлены или очищены, в то время как другие, такие как шифрование, доступны только для чтения и могут быть только просмотрены. Поддержка определенных атрибутов зависит от используемой файловой системы.

##### **Метки времени (timestamps)**

Доступ (Access - last access) — время, когда файл был прочитан последний раз. Это время меняется при доступе таких системных вызовов как mknod(2), utimes(2) и read(2). Если это текстовый файл, то дата последнего доступа обновляется при каждом его открытии. Если это исполнимый файл, то дата доступа обновится при его запуске. При некоторых опциях монтирования диска (noatime или relatime) это значение может быть неточным. Модифицирован (Modify - last modified) — время последнего изменения содержимого файла. То есть если это текстовый файл, то время модификации поменяется когда вы его откроете и удалите какое-то слово или что-то допишите. Меняется системными вызовами mknod(2), utimes(2) и write(2). Изменён (Change - last changed) — Время последнего изменения метаданных файлов в файловой системе. То есть если в файле изображения вы измените EXIF метаданные — это будет модификация (поскольку по сути поменяется содержимое файла). Примером Изменения файла является смена разрешений доступа к нему (чтение, запись, выполнение), смена владельца, группы и т. д. Меняется с chmod(2), chown(2), link(2), mknod.

Минимум знаний на 3.5 балла (для студентов БГТУ)

Определение операционной системы

Использование стека при вызове процедур и при возникновении прерываний

.dll-модули.

Переключение контекста потока.

Механизмы синхронизации: критические секции, мьютексы, семафоры, события, барьеры, атомарные операции. Отличия друг от друга и сценарии использования.

Адресное пространство процесса.

Виртуальная память (интерпретация всех столбцов в выводе диспетчера задач в Windows и top и free в Linux.

Использование стека при вызове процедур и при возникновении прерываний.

SID и ACL в Windows.

rwxrwxrwx в \*nix.