Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Факультет физики и информационных технологий
Кафедра автоматизированных систем обработки информации

СОГЛАСОВАНО
Заведующий кафедрой
автоматизированных систем
обработки информации
А.В.Воруев
__ _____ 2023 г.

СОГЛАСОВАНО
Декан
факультета физики и
информационных технологий
Д.Л.Коваленко
__ _____ 2023 г.

ЭЛЕКТРОННЫЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

ОПЕРАЦИОННЫЕ СИСТЕМЫ

для специальности
1-53 01 02 Автоматизированные системы обработки информации

составители: заведующий кафедрой АСОИ, к.т.н., доцент, Воруев А.В.
старший преподаватель Кучеров А.И.
старший преподаватель Дробышевский В.А.

Рассмотрено и утверждено
на заседании кафедры АСОИ
__14 марта___ 2023 г., протокол № _8_

Рассмотрено и утверждено
на заседании научно-методического
совета университета
_____ 2023 г., протокол № ___

Гомель 2023

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Электронный учебно-методический комплекс (ЭУМК) по дисциплине «Операционные системы» представляет собой комплекс систематизированных учебных, методических и вспомогательных материалов, предназначенных для использования в образовательном процессе специальности 1-53 01 02 – Автоматизированные системы обработки информации.

ЭУМК разработан в соответствии со следующими нормативными документами:

1. Положением об учебно-методическом комплексе на уровне высшего образования, утвержденном постановлением Министерства образования Республики Беларусь от 26.07.2011 №167.

2. Учебного плана УВО специальности 1-53 01 02 Автоматизированные системы обработки информации, регистрационный № I 53-1-21/УП, дата утверждения 31.05.2021.

3. Учебной программой по учебной дисциплине «Операционные системы» для специальности 1-53 01 02 Автоматизированные системы обработки информации, утвержденной 28.07.2021, регистрационный номер УД-2021-92/уч.

Цель создания ЭУМК – обеспечить приобретение теоретических знаний и практических навыков при подготовке специалистов в области практического применения технологий программного управления вычислительной техникой.

ЭУМК направлен на всестороннюю подготовку студентов теоретическим основам и практическим навыкам по управлению операционными системами и процессами, которые работают в их среде. Отдельное внимание уделяется вопросам виртуализации и сетевого взаимодействия. Организация изучения дисциплины на основе ЭУМК предполагает продуктивную образовательную деятельность, позволяющую сформировать социально-личностные и профессиональные компетенции будущих специалистов.

ЭУМК способствует успешному осуществлению учебной деятельности, дает возможность планировать и осуществлять самостоятельную управляемую работу студентов, обеспечивает рациональное распределение учебного времени по темам учебной дисциплины и совершенствование методики проведения занятий.

ЭУМК состоит из теоретического, практического и вспомогательного разделов. Теоретический раздел содержит тексты лекций. Практический раздел содержит методические рекомендации к лабораторным работам, тестовые задания и вопросы для самоконтроля. Вспомогательный раздел содержит учебную программу и список литературы.

Теоретический раздел содержит лекционный материал по всем темам учебной программы, включая и темы, вынесенные на самостоятельное изучение. В разделе так же содержатся рекомендации по организации и выполнению управляемой самостоятельной работы по трем уровням сложности.

Практический раздел включает в себя темы лабораторных занятий и задания с краткими методическими указаниями по выполнению лабораторных работ. В разделе так же приводятся некоторый набор тестовых заданий и к каждой теми указаны вопросы для самоконтроля.

Вспомогательный раздел содержит необходимые элементы учебно-программной документации по дисциплине с указанием рекомендуемой литературы (основной, дополнительной, вспомогательной).

Все разделы ЭУМК в полной мере соответствуют содержанию учебной программы и объему учебного плана.

Дисциплина государственного компонента «Операционные системы» изучается студентами 1 курса дневной, заочной и дистанционной форм обучения специальности 1-53 01 02 - «Автоматизированные системы обработки информации».

Дневная форма обучения: всего часов по плану – 104 (3 зач. ед.); аудиторное количество часов – 52, из них: лекции – 36, лабораторные занятия – 16. Курсовой проект.

Форма отчётности – зачет во 2 семестре.

Заочная, заочная сокращенная и дистанционная формы обучения: всего часов по плану – 104, аудиторное количество часов – 14, из них: лекции – 10, лабораторные занятия – 4. Курсовой проект.

Форма отчётности – контрольная работа и зачет во 2 семестре.

2 ТЕКСТЫ ЛЕКЦИЙ

Раздел 1 Операционные системы в составе инструментов системного программного обеспечения

Тема 1.1 Назначение операционных систем

Определение: Под операционной системой понимается набор управляющих программ, предназначенных для управления ресурсами вычислительной системы.

Операционная система является фундаментальным компонентом системного программного обеспечения. Слой ОС основной предмет настоящего курса лекций. Операционная система компьютера представляет собой комплекс взаимосвязанных программ, который действует как интерфейс между приложениями и пользователями, с одной стороны, и аппаратурой компьютера – с другой.

В соответствии с этим определением ОС выполняет две группы функций:

предоставление пользователю или программисту вместо реальной аппаратуры компьютера расширенной виртуальной машины, с которой удобней работать и которую легче программировать;

повышение эффективности использования компьютера путем рационального управления его ресурсами в соответствии с некоторым критерием.

ОС как виртуальная машина. Для того чтобы успешно решать свои задачи, современный пользователь или даже прикладной программист может обойтись без досконального знания аппаратного устройства компьютера. Ему не обязательно быть в курсе того, как функционируют различные электронные блоки и электромеханические узлы компьютера. Такие частности, как используемая при записи частотная модуляция или текущее состояние двигателя механизма перемещения магнитных головок чтения/записи, не должны волновать программиста. Именно операционная система скрывает от программиста большую часть особенностей аппаратуры и предоставляет возможность простой и удобной работы с требуемыми файлами.

В результате реальная машина, способная выполнять только небольшой набор элементарных действий, определяемых ее системой команд, превращается в виртуальную машину, выполняющую широкий набор гораздо более мощных функций. Виртуальная машина тоже управляется командами, но это уже команды другого, более высокого уровня, такие, как удалить файл с определенным именем, запустить на выполнение некоторую прикладную программу, повысить приоритет задачи, вывести текст из файла на печать. Таким образом, назначение ОС состоит в предоставлении пользователю/программисту некоторой расширенной виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальный компьютер или реальную сеть.

ОС как система управления ресурсами. Операционная система не только предоставляет пользователям и программистам удобный интерфейс к аппаратным средствам компьютера, но и является механизмом, распределяющим ресурсы компьютера.

К основным ресурсам современных вычислительных систем могут быть отнесены такие ресурсы, как процессоры, основная память, таймеры, наборы данных, диски, накопители на магнитных лентах, принтеры, сетевые устройства и некоторые другие. Ресурсы распределяются между процессами. Процесс (задача) представляет собой базовое понятие большинства современных ОС и часто кратко определяется как программа в стадии выполнения. Программа – это статический объект, представляющий собой файл с кодами и данными. Процесс – это динамический объект, который возникает в операционной системе после того, как пользователь или сама операционная система решает «запустить программу на выполнение», то есть создать новую единицу вычислительной работы.

Управление ресурсами вычислительной системы с целью наиболее эффективного их использования является назначением операционной системы.

Управление ресурсами включает решение следующих общих задач, не зависящих от типа ресурса:

- планирование ресурса – то есть определение, какому процессу, когда и в каком количестве (если ресурс может выделяться частями) следует выделить данный ресурс;
- удовлетворение запросов на ресурсы;
- отслеживание состояния и учет использования ресурса – то есть поддержание оперативной информации о том, занят или свободен ресурс и какая доля ресурса уже распределена;
- разрешение конфликтов между процессами.

Управление ресурсами составляет важную часть функций любой операционной системы, в особенности мультипрограммной. Многие функции управления ресурсами выполняются операционной системой автоматически и прикладному программисту недоступны.

Тема 1.2 Интерфейсы операционных систем

Пользовательским интерфейсом называется набор приемов взаимодействия пользователя с приложением. Пользовательский интерфейс включает общение пользователя с приложением и язык общения. Пакетный интерфейс. Консольный ввод. Командный интерфейс. Простой графический интерфейс. Псевдографические системы. WIMP-интерфейс. WEB-интерфейс. Речевой интерфейс. SILK-системы. Интерфейс распознавания образов. Тактильный интерфейс. Семантический интерфейс.

Интерфейс прикладного программирования. Прикладные программисты используют в своих приложениях обращения к ОС, когда для выполнения тех или иных действий им требуется особый статус, которым обладает только операционная система. Например, в большинстве современных ОС все действия, связанные с управлением аппаратными средствами компьютера, может выполнять только ОС. Помимо этих функций, прикладной программист может воспользоваться набором сервисных функций ОС, которые упрощают написание приложений. Функции такого типа реализуют универсальные действия, часто требующиеся в различных приложениях, такие, например, как обработка текстовых строк. Эти функции могли бы быть выполнены и самим приложением, однако гораздо проще использовать уже готовые, отлаженные процедуры, включенные в состав операционной системы. В то же время даже при наличии в ОС соответствующей функции программист может реализовать ее самостоятельно в рамках приложения, если предложенный операционной системой вариант его не вполне устраивает.

Возможности операционной системы доступны прикладному программисту в виде набора функций, называющегося интерфейсом прикладного программирования (Application Programming Interface, API). От конечного пользователя эти функции скрыты за оболочкой алфавитно-цифрового или графического пользовательского интерфейса.

Для разработчиков приложений все особенности конкретной операционной системы представлены особенностями ее API. Поэтому операционные системы с различной внутренней организацией, но с одинаковым набором функций API кажутся им одной и той же ОС, что упрощает стандартизацию операционных систем и обеспечивает переносимость приложений между внутренне различными ОС, соответствующими определенному стандарту на API. Например, следование общим стандартам API UNIX, одним из которых является стандарт POSIX (Portable Operating System Interface for Computer Environment), позволяет говорить о некоторой обобщенной операционной системе UNIX, хотя многочисленные версии этой ОС от разных производителей иногда существенно отличаются внутренней организацией.

Приложения выполняют обращения к функциям API с помощью системных вызовов. Способ, которым приложение получает услуги операционной системы, очень похож на вызов

подпрограмм. Информация, нужная ОС и состоящая обычно из идентификатора команды и данных, помещается в определенное место памяти, в регистры и/или стек. Затем управление передается операционной системе, которая выполняет требуемую функцию и возвращает результаты через память, регистры или стеки. Если операция проведена неуспешно, то результат включает индикацию ошибки.

Способ реализации системных вызовов зависит от структурной организации ОС, которая, в свою очередь, тесно связана с особенностями аппаратной платформы. Кроме того, он зависит от языка программирования. При использовании языков высокого уровня функции ОС вызываются тем же способом, что и написанные пользователем подпрограммы, требуя задания определенных аргументов в определенном порядке.

Пользовательский интерфейс. Операционная система должна обеспечивать удобный интерфейс не только для прикладных программ, но и для человека, работающего за терминалом. Этот человек может быть конечным пользователем, администратором ОС или программистом.

Современные ОС поддерживают развитые функции пользовательского интерфейса для интерактивной работы за терминалами двух типов: алфавитно-цифровыми и графическими.

При работе за алфавитно-цифровым терминалом пользователь имеет в своем распоряжении систему команд, мощность которой отражает функциональные возможности данной ОС. Обычно командный язык ОС позволяет запускать и останавливать приложения, выполнять различные операции с файлами и каталогами, получать информацию о состоянии ОС (количество работающих процессов, объем свободного пространства на дисках и т. п.), администрировать систему. Команды могут вводиться не только в интерактивном режиме с терминала, но и считываться из так называемого командного файла, содержащего некоторую последовательность команд. Программный модуль ОС, ответственный за чтение отдельных команд или же последовательности команд из командного файла, иногда называют командным интерпретатором.

Ввод команды может быть упрощен, если операционная система поддерживает графический пользовательский интерфейс. В этом случае пользователь для выполнения нужного действия с помощью мыши выбирает на экране нужный пункт меню или графический символ.

Раздел 2 Формальный подход к описанию операционных систем

Тема 2.1 Функции операционных систем

Функции операционной системы компьютера обычно группируются либо в соответствии с типами локальных ресурсов, которыми управляет ОС, либо в соответствии со специфическими задачами, применимыми ко всем ресурсам. Иногда такие группы функций называют подсистемами. Наиболее важными подсистемами управления ресурсами являются подсистемы управления процессами, памятью, файлами и внешними устройствами, а подсистемами, общими для всех ресурсов, являются подсистемы пользовательского интерфейса, защиты данных и администрирования.

Управление процессами. Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами.

Для каждого вновь создаваемого процесса ОС генерирует системные информационные структуры, которые содержат данные о потребностях процесса в ресурсах вычислительной системы, а также о фактически выделенных ему ресурсах.

Чтобы процесс мог быть выполнен, операционная система должна назначить ему область оперативной памяти, в которой будут размещены коды и данные процесса, а также предоставить ему необходимое количество процессорного времени. Кроме того, процессу может понадобиться доступ к таким ресурсам, как файлы и устройства ввода-вывода.

В мультипрограммной операционной системе одновременно может существовать несколько процессов. Часть процессов порождается по инициативе пользователей и их приложений. Такие процессы обычно называют пользовательскими. Другие процессы, называемые системными, инициализируются самой операционной системой для выполнения своих функций.

Поскольку процессы часто одновременно претендуют на одни и те же ресурсы, то в обязанности ОС входит поддержание очередей заявок процессов на ресурсы, например, очереди к процессору, к принтеру, к последовательному порту.

Важная задача операционной системы – защита ресурсов, выделенных данному процессу, от остальных процессов. Одним из наиболее тщательно защищаемых ресурсов процесса являются области оперативной памяти, в которой хранятся коды и данные процесса. Совокупность всех областей оперативной памяти, выделенных операционной системой процессу, называется его адресным пространством. Защищаются и другие типы ресурсов, такие, как файлы, внешние устройства и т. д. Операционная система может не только защищать ресурсы, выделенные одному процессу, но и организовывать их совместное использование, например, разрешать доступ к некоторой области памяти нескольким процессам.

На протяжении периода существования процесса его выполнение может быть многократно прервано и продолжено. Для того чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды идентифицируется состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок выполняемых данным процессом системных вызовов и т. д. Эта информация называется контекстом процесса. Говорят, что при смене процесса происходит переключение контекстов.

Таким образом, подсистема управления процессами планирует выполнение процессов, то есть распределяет процессорное время между несколькими одновременно существующими в системе процессами, занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами.

Тема 2.2 Процессы и потоки

Важнейшей функцией операционной системы является организация рационального использования всех ее аппаратных и информационных ресурсов. К основным ресурсам могут быть отнесены процессоры, память, внешние устройства, данные и программы. Располагающая одними и теми же аппаратными ресурсами, но управляемая различными ОС, вычислительная система может работать с разной степенью эффективности. Поэтому знание внутренних механизмов операционной системы позволяет косвенно судить о ее эксплуатационных возможностях и характеристиках. Хотя и в однопрограммной ОС необходимо решать задачи управления ресурсами (например, распределение памяти между приложением и ОС), главные сложности на этом пути возникают в мультипрограммных ОС, в которых за ресурсы конкурируют сразу несколько приложений. Именно поэтому большая часть всех проблем относится к мультипрограммным системам.

Одной из основных подсистем мультипрограммной ОС, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами и потоками, которая занимается их созданием и уничтожением, поддерживает взаимодействие между ними, а также распределяет процессорное время между несколькими одновременно существующими в системе процессами и потоками.

Каждый раз, когда процесс завершается, ОС предпринимает шаги, чтобы «зачистить следы» его пребывания в системе. Подсистема управления процессами закрывает все файлы, с которыми работал процесс, освобождает области оперативной памяти, отведенные под коды, данные и системные информационные структуры процесса. Выполняется коррекция

всевозможных очередей ОС и списков ресурсов, в которых имелись ссылки на завершаемый процесс.

Чтобы поддерживать мультипрограммирование, ОС должна определить и оформить для себя те внутренние единицы работы, между которыми будут разделяться процессор и другие ресурсы компьютера. В настоящее время в большинстве операционных систем определены два типа единиц работы. Более крупная единица работы, обычно носящая название процесса или задачи, требует для своего выполнения нескольких более мелких работ, для обозначения которых используют термины «поток».

Очевидно, что любая работа вычислительной системы заключается в выполнении некоторой программы. Поэтому и с процессом, и с потоком связывается определенный программный код, который для этих целей оформляется в виде исполняемого модуля.

В операционных системах процесс рассматривается ОС как заявка на потребление всех видов ресурсов, кроме одного процессорного времени. Этот последний важнейший ресурс распределяется операционной системой между другими единицами работы потоками, которые и получили свое название благодаря тому, что они представляют собой последовательности (потоки выполнения) команд. В простейшем случае процесс состоит из одного потока, и именно таким образом трактовалось понятие «процесс» до середины 80-х годов (например, в ранних версиях UNIX) и в таком же виде оно сохранилось в некоторых современных ОС. В таких системах понятие «поток» полностью поглощается понятием «процесс», то есть остается только одна единица работы и потребления ресурсов – процесс. Мультипрограммирование осуществляется в таких ОС на уровне процессов.

Для того чтобы процессы не могли вмешаться в распределение ресурсов, а также не могли повредить коды и данные друг друга, важнейшей задачей ОС является изоляция одного процесса от другого. Для этого операционная система обеспечивает каждый процесс отдельным виртуальным адресным пространством, так что ни один процесс не может получить прямого доступа к командам и данным другого процесса.

При необходимости взаимодействия процессы обращаются к операционной системе, которая, выполняя функции посредника, предоставляет им средства межпроцессной связи конвейеры, почтовые ящики, разделяемые секции памяти и некоторые другие средства.

Однако в системах, в которых отсутствует понятие потока, возникают проблемы при организации параллельных вычислений в рамках процесса. А такая необходимость может возникать. Приложение, выполняемое в рамках одного процесса, может обладать внутренним параллелизмом, который в принципе мог бы позволить ускорить его решение. Если, например, в программе предусмотрено обращение к внешнему устройству, то на время этой операции можно не блокировать выполнение всего процесса, а продолжить вычисления по другой ветви программы. Параллельное выполнение нескольких работ в рамках одного интерактивного приложения повышает эффективность работы пользователя.

В общем случае использование для создания процессов стандартных средств ОС не позволяет учесть тот факт, что эти процессы решают единую задачу, а значит, имеют много общего между собой они могут работать с одними и теми же данными, использовать один и тот же кодовый сегмент, наделяться одними и теми же правами доступа к ресурсам вычислительной системы.

Таким образом, в операционной системе наряду с процессами нужен другой механизм распараллеливания вычислений, который учитывал бы тесные связи между отдельными ветвями вычислений одного и того же приложения. Для этих целей современные ОС предлагают механизм многопоточной обработки (multithreading). При этом вводится новая единица работы поток выполнения, а понятие «процесс» в значительной степени меняет смысл. Понятию «поток» соответствует последовательный переход процессора от одной команды программы к другой. ОС распределяет процессорное время между потоками. Процессу ОС назначает адресное пространство и набор ресурсов, которые совместно используются всеми его потоками.

Создание потоков требует от ОС меньших накладных расходов, чем процессов. В отличие от процессов, которые принадлежат разным конкурирующим приложениям, все потоки одного процесса всегда принадлежат одному приложению, поэтому ОС изолирует потоки в гораздо меньшей степени, нежели процессы в традиционной мультипрограммной системе. Все потоки одного процесса используют общие файлы, таймеры, устройства, одну и ту же область оперативной памяти, одно и то же адресное пространство и разделяют одни и те же глобальные переменные. Поскольку каждый поток может иметь доступ к любому виртуальному адресу процесса, один поток может использовать стек другого потока. Между потоками одного процесса нет полной защиты, потому что, во-первых, это невозможно, а во-вторых, не нужно. Чтобы организовать взаимодействие и обмен данными, потокам вовсе не требуется обращаться к ОС, им достаточно использовать общую память один поток записывает данные, а другой читает их. С другой стороны, потоки разных процессов по-прежнему хорошо защищены друг от друга.

Итак, мультипрограммирование более эффективно на уровне потоков, а не процессов. Наибольший эффект от введения многопоточной обработки достигается в мультипроцессорных системах, в которых потоки, в том числе и принадлежащие одному процессу, могут выполняться на разных процессорах действительно параллельно (а не псевдопараллельно).

Тема 2.3 Алгоритмы планирования

Алгоритмы планирования, основанные на квантовании.

В основе многих вытесняющих алгоритмов планирования лежит концепция квантования. В соответствии с этой концепцией каждому потоку поочередно для выполнения предоставляется ограниченный непрерывный период процессорного времени квант.

Смена активного потока происходит, если:

- поток завершился и покинул систему;
- произошла ошибка;
- поток перешел в состояние ожидания;
- исчерпан квант процессорного времени, отведенный данному потоку.

Поток, который исчерпал свой квант, переводится в состояние готовности и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый поток из очереди готовых.

Кванты, выделяемые потокам, могут быть одинаковыми для всех потоков или различными.

Если квант короткий, то суммарное время, которое проводит поток в ожидании процессора, прямо пропорционально времени, требуемому для его выполнения (то есть времени, которое потребовалось бы для выполнения этого потока при монопольном использовании вычислительной системы).

Чем больше квант, тем выше вероятность того, что потоки завершатся в результате первого же цикла выполнения, и тем менее явной становится зависимость времени ожидания потоков от их времени выполнения. При достаточно большом кванте алгоритм квантования вырождается в алгоритм последовательной обработки, присущий однопрограммным системам, при котором время ожидания задачи в очереди вообще никак не зависит от ее длительности. Кванты, выделяемые одному потоку, могут быть фиксированной величины, а могут и изменяться в разные периоды жизни потока. Пусть, например, первоначально каждому потоку назначается достаточно большой квант, а величина каждого следующего кванта уменьшается до некоторой заранее заданной величины. В таком случае преимущество получают короткие задачи, которые успевают выполняться в течение первого кванта, а длительные вычисления будут проводиться в фоновом режиме. Можно представить себе алгоритм планирования, в котором каждый следующий квант, выделяемый определенному потоку, больше предыдущего. Такой подход позволяет уменьшить накладные расходы на

переключение задач в том случае, когда сразу несколько задач выполняют длительные вычисления.

Многозадачные ОС теряют некоторое количество процессорного времени для выполнения вспомогательных работ во время переключения контекстов задач. При этом запоминаются и восстанавливаются регистры, флаги и указатели стека, а также проверяется статус задач для передачи управления. Затраты на эти вспомогательные действия не зависят от величины кванта времени, поэтому чем больше квант, тем меньше суммарные накладные расходы, связанные с переключением потоков.

В алгоритмах, основанных на квантовании, какую бы цель они не преследовали (предпочтение коротких или длинных задач, минимизация накладных расходов, связанных с переключениями), не используется никакой предварительной информации о задачах. При поступлении задачи на обработку ОС не имеет никаких сведений о том, является ли она короткой или длинной, насколько интенсивными будут ее запросы к устройствам ввода-вывода, насколько важно ее быстрое выполнение и т. д. Дифференциация обслуживания при квантовании базируется на «истории существования» потока в системе.

Алгоритмы планирования, основанные на приоритетах.

Другой важной концепцией, лежащей в основе многих вытесняющих алгоритмов планирования, является приоритетное обслуживание. Приоритетное обслуживание предполагает наличие у потоков некоторой изначально известной характеристики приоритета, на основании которой определяется порядок их выполнения. Приоритет это число, характеризующее степень привилегированности потока при использовании ресурсов вычислительной машины, в частности, процессорного времени: чем выше приоритет, тем выше привилегии, тем меньше времени будет проводить поток в очередях.

Приоритет может выражаться целым или дробным, положительным или отрицательным значением. В некоторых ОС принято считать, что приоритет потока тем выше, чем больше (в арифметическом смысле) число, обозначающее приоритет. В других системах, наоборот, чем меньше число, тем выше приоритет. В большинстве операционных систем, поддерживающих потоки, приоритет потока непосредственно связан с приоритетом процесса, в рамках которого выполняется данный поток. Приоритет процесса назначается операционной системой при его создании.

Во многих ОС предусматривается возможность изменения приоритетов в течение жизни потока. Изменение приоритета может происходить по инициативе самого потока, когда он обращается с соответствующим вызовом к операционной системе, или по инициативе пользователя, когда он выполняет соответствующую команду. Кроме того, ОС сама может изменять приоритеты потоков в зависимости от ситуации, складывающейся в системе. В последнем случае приоритеты называются динамическими в отличие от неизменяемых, фиксированных, приоритетов.

При создании процесс в зависимости от класса получает по умолчанию базовый приоритет в верхней или нижней части диапазона. Базовый приоритет процесса в дальнейшем может быть повышен или понижен операционной системой. Первоначально поток получает значение базового приоритета из диапазона базового приоритета процесса, в котором он был создан. Пусть, например, значение базового приоритета некоторого процесса равно К. Тогда все потоки данного процесса получат базовые приоритеты из диапазона [К-2, К+2]. Отсюда видно, что, изменяя базовый приоритет процесса, ОС может влиять на базовые приоритеты его потоков.

В Windows NT с течением времени приоритет потока, относящегося к классу потоков с переменными приоритетами, может отклоняться от базового приоритета потока, причем эти изменения могут быть не связаны с изменениями базового приоритета процесса. ОС может повышать приоритет потока (который в этом случае называется динамическим) в тех случаях, когда поток не полностью использовал отведенный ему квант, или понижать приоритет, если квант был использован полностью. ОС наращивает приоритет дифференцированно в зависимости от того, какого типа событие не дало потоку полностью использовать квант. В

частности, ОС повышает приоритет в большей степени потокам, которые ожидают ввода с клавиатуры (интерактивным приложениям), и в меньшей степени потокам, выполняющим дисковые операции. Именно на основе динамических приоритетов осуществляется планирование потоков. Начальной точкой отсчета для динамического приоритета является значение базового приоритета потока. Значение динамического приоритета потока ограничено снизу его базовым приоритетом, верхней же границей является нижняя граница диапазона приоритетов реального времени.

В современных ОС существуют две разновидности приоритетного планирования: обслуживание с относительными приоритетами и обслуживание с абсолютными приоритетами.

В обоих случаях выбор потока на выполнение из очереди готовых осуществляется одинаково: выбирается поток, имеющий наивысший приоритет. Однако проблема определения момента смены активного потока решается по-разному. В системах с относительными приоритетами активный поток выполняется до тех пор, пока он сам не покинет процессор, перейдя в состояние ожидания (или же произойдет ошибка, или поток завершится).

В системах с абсолютными приоритетами выполнение активного потока прерывается, кроме указанных выше причин, еще при одном условии: если в очереди готовых потоков появился поток, приоритет которого выше приоритета активного потока. В этом случае прерванный поток переходит в состояние готовности.

В системах, в которых планирование осуществляется на основе относительных приоритетов, минимизируются затраты на переключения процессора с одной работы на другую. С другой стороны, здесь могут возникать ситуации, когда одна задача занимает процессор долгое время. Ясно, что для систем разделения времени и реального времени такая дисциплина обслуживания не подходит: интерактивное приложение может ждать своей очереди часами, пока вычислительной задаче не потребуется ввод-вывод. А вот в системах пакетной обработки (в том числе известной ОС OS/360) относительные приоритеты используются широко.

В системах с абсолютными приоритетами время ожидания потока в очередях может быть сведено к минимуму, если ему назначить самый высокий приоритет. Такой поток будет вытеснять из процессора все остальные потоки (кроме потоков, имеющих такой же наивысший приоритет). Это делает планирование на основе абсолютных приоритетов подходящим для систем управления объектами, в которых важна быстрая реакция на событие.

Смешанные алгоритмы планирования.

Во многих операционных системах алгоритмы планирования построены с использованием как концепции квантования, так и приоритетов. Например, в основе планирования лежит квантование, но величина кванта и/или порядок выбора потока из очереди готовых определяется приоритетами потоков. Именно так реализовано планирование в системе Windows NT, в которой квантование сочетается с динамическими абсолютными приоритетами. На выполнение выбирается готовый поток с наивысшим приоритетом. Ему выделяется квант времени. Если во время выполнения в очереди готовых появляется поток с более высоким приоритетом, то он вытесняет выполняемый поток. Вытесненный поток возвращается в очередь готовых, причем он становится впереди всех остальных потоков, имеющих такой же приоритет.

Рассмотрим более подробно алгоритм планирования в операционной системе UNIX System V Release 4. В этой ОС понятие «поток» отсутствует, и планирование осуществляется на уровне процессов. В системе UNIX System V Release 4 реализована вытесняющая многозадачность, основанная на использовании приоритетов и квантования.

Каждый процесс в зависимости от задачи, которую он решает, относится к одному из трех определенных в системе приоритетных классов: классу реального времени, классу системных процессов или классу процессов разделения времени. Назначение и обработка приоритетов выполняются для разных классов по-разному. Процессы системного класса,

зарезервированные для ядра, используют стратегию фиксированных приоритетов. Уровень приоритета процессу назначается ядром и никогда не изменяется.

Процессы реального времени также используют стратегию фиксированных приоритетов, но пользователь может их изменять. Так как при наличии готовых к выполнению процессов реального времени другие процессы не рассматриваются, то процессы реального времени надо тщательно проектировать, чтобы они не захватывали процессор на слишком долгое время. Характеристики планирования процессов реального времени включают две величины: уровень глобального приоритета и квант времени. Для каждого уровня приоритета по умолчанию имеется своя величина кванта времени. Процессу разрешается захватывать процессор на указанный квант времени, а по его истечении планировщик снимает процесс с выполнения.

Процессы разделения времени были до появления UNIX System V Release 4 единственным классом процессов, и по умолчанию UNIX System V Release 4 назначает новому процессу именно этот класс. Состав класса процессов разделения времени наиболее неопределенный и часто меняющийся в отличие от системных процессов и процессов реального времени. Для справедливого распределения времени процессора между процессами в этом классе используется стратегия динамических приоритетов. Величина приоритета, назначаемого процессам разделения времени, вычисляется пропорционально значениям двух составляющих: пользовательской части и системной части. Пользовательская часть приоритета может быть изменена администратором и владельцем процесса, но в последнем случае только в сторону его снижения.

Системная составляющая позволяет планировщику управлять процессами в зависимости от того, как долго они занимают процессор, не уходя в состояние ожидания. У тех процессов, которые потребляют большие периоды процессорного времени без ухода в состояние ожидания, приоритет снижается, а у тех процессов, которые часто уходят в состояние ожидания после короткого периода использования процессора, приоритет повышается. Таким образом, процессам, ведущим себя не «по-джентельменски», дается низкий приоритет. Это означает, что они реже выбираются для выполнения. Это ущемление в правах компенсируется тем, что процессам с низким приоритетом даются большие кванты времени, чем процессам с высокими приоритетами. Таким образом, хотя низкоприоритетный процесс и не работает так часто, как высокоприоритетный, но зато, когда он наконец выбирается для выполнения, ему отводится больше времени.

Мультипрограммирование на основе прерываний.

Основная цель введения прерываний — реализация асинхронного режима функционирования и распараллеливание работы отдельных устройств вычислительного комплекса.

Прерывания представляют собой механизм, позволяющий координировать параллельное функционирование отдельных устройств вычислительной системы и реагировать на особые состояния, возникающие при работе процессора, то есть прерывание — это принудительная передача управления от выполняемой программы к системе (а через нее — к соответствующей программе обработки прерывания), происходящая при возникновении определенного события.

Прерывания являются основной движущей силой любой операционной системы. Отключите систему прерываний — и «жизнь» в операционной системе немедленно остановится. Периодические прерывания от таймера вызывают смену процессов в мультипрограммной ОС, а прерывания от устройств ввода-вывода управляют потоками данных, которыми вычислительная система обменивается с внешним миром.

Как верно было замечено: «Прерывания названы так весьма удачно, поскольку они прерывают нормальную работу системы». Другими словами, система прерываний переводит процессор на выполнение потока команд, отличного от того, который выполнялся до сих пор, с последующим возвратом к исходному коду. Из сказанного можно сделать вывод о том, что механизм прерываний очень похож на механизм выполнения процедур. Это на самом деле

так, хотя между этими механизмами имеется важное отличие. Переключение по прерыванию отличается от переключения, которое происходит по команде безусловного или условного перехода, предусмотренной программистом в потоке команд приложения. Переход по команде происходит в заранее определенных программистом точках программы в зависимости от исходных данных, обрабатываемых программой. Прерывание же происходит в произвольной точке потока команд программы, которую программист не может прогнозировать. Прерывание возникает либо в зависимости от внешних по отношению к процессу выполнения программы событий, либо при появлении непредвиденных аварийных ситуаций в процессе выполнения данной программы. Сходство же прерываний с процедурами состоит в том, что в обоих случаях выполняется некоторая подпрограмма, обрабатывающая специальную ситуацию, а затем продолжается выполнение основной ветви программы.

Механизм прерываний реализуется аппаратно-программными средствами. Структуры систем прерывания (в зависимости от аппаратной архитектуры) могут быть самыми разными, но все они имеют одну общую особенность — прерывание непременно влечет за собой изменение порядка выполнения команд процессором.

Тема 2.4 Синхронизация процессов и потоков

Синхронизация процессов и потоков. Цели и средства синхронизации.

Существует достаточно обширный класс средств операционной системы, с помощью которых обеспечивается взаимная синхронизация процессов и потоков. Потребность в синхронизации потоков возникает только в мультипрограммной операционной системе и связана с совместным использованием аппаратных и информационных ресурсов вычислительной системы. Во многих операционных системах эти средства называются средствами межпроцессного взаимодействия – InterProcess Communications (IPC), что отражает историческую первичность понятия «процесс» по отношению к понятию «поток». Обычно к средствам IPC относят не только средства межпроцессной синхронизации, но и средства межпроцессного обмена данными.

Потоки в общем случае (когда программист не предпринял специальных мер по их синхронизации) протекают независимо, асинхронно друг другу. Это справедливо как по отношению к потокам одного процесса, выполняющим общий программный код, так и по отношению к потокам разных процессов, каждый из которых выполняет собственную программу. Любое взаимодействие процессов или потоков связано с их синхронизацией, которая заключается в согласовании их скоростей путем приостановки потока до наступления некоторого события и последующей его активизации при наступлении этого события. Синхронизация лежит в основе любого взаимодействия потоков, связано ли это взаимодействие с разделением ресурсов или с обменом данными.

Необходимость синхронизации и гонки.

Пренебрежение вопросами синхронизации в многопоточной системе может привести к неправильному решению задачи или даже к краху системы. Рассмотрим, например, задачу ведения базы данных клиентов некоторого предприятия. Каждому клиенту отводится отдельная запись в базе данных, в которой среди прочих полей имеются поля Заказ и Оплата. Программа, ведущая базу данных, оформлена как единый процесс, имеющий несколько потоков, в том числе поток А, который заносит в базу данных информацию о заказах, поступивших от клиентов, и поток В, который фиксирует в базе данных сведения об оплате клиентами выставленных счетов. Оба эти потока совместно работают над общим файлом базы данных, используя однотипные алгоритмы, включающие три шага.

1. Считать из файла базы данных в буфер запись о клиенте с заданным идентификатором.
2. Внести новое значение в поле Заказ (для потока А) или Оплата (для потока В).
3. Вернуть модифицированную запись в файл базы данных.

Тема 2.5 Тупиковые ситуации

Важным понятием синхронизации потоков является понятие «критической секции» программы. Критическая секция — это часть программы, результат выполнения которой может непредсказуемо меняться, если переменные, относящиеся к .этой части программы, изменяются другими потоками в то время, когда выполнение этой части еще не завершено. Критическая секция всегда определяется по отношению к определенным критическим данным, при несогласованном изменении которых могут возникнуть нежелательные эффекты. В предыдущем примере такими критическими данными являлись записи файла базы данных. Во всех потоках, работающих с критическими данными, должна быть определена критическая секция. Заметим, что в разных потоках критическая секция состоит в общем случае из разных последовательностей команд.

Чтобы исключить эффект гонок по отношению к критическим данным, необходимо обеспечить, чтобы в каждый момент времени в критической секции, связанной с этими данными, находился только один поток. При этом неважно, находится этот поток в активном или в приостановленном состоянии. Этот прием называют взаимным исключением. Операционная система использует разные способы реализации взаимного исключения. Некоторые способы пригодны для взаимного исключения при вхождении в критическую секцию только потоков одного процесса, в то время как другие могут обеспечить взаимное исключение и для потоков разных процессов.

Самый простой и в то же время самый неэффективный способ обеспечения Взаимного исключения состоит в том, что операционная система позволяет потоку запрещать любые прерывания на время его нахождения в критической секции. Однако этот способ практически не применяется, так как опасно доверять управление системой пользовательскому потоку — он может надолго занять процессор, а при крахе потока в критической секции крах потерпит вся система, потому что прерывания никогда не будут разрешены.

Для синхронизации потоков одного процесса прикладной программист может использовать глобальные блокирующие переменные. С этими переменными, к которым все потоки процесса имеют прямой доступ, программист работает, не обращаясь к системным вызовам ОС.

Каждому набору критических данных ставится в соответствие двоичная переменная, которой поток присваивает значение 0, когда он входит в критическую секцию, и значение 1, когда он ее покидает. Перед входом в критическую секцию поток проверяет, не работает ли уже какой-нибудь поток с данными D. Если переменная F(D) установлена в 0, то данные заняты и проверка циклически повторяется. Если же данные свободны (F(D) =1), то значение переменной F(D) устанавливается в 0 и поток входит в критическую секцию. После того как поток выполнит все действия с данными D, значение переменной F(D) :нова устанавливается равным 1.

Блокирующие переменные могут использоваться не только при доступе к разделяемым данным, но и при доступе к разделяемым ресурсам любого вида.

Если все потоки написаны с учетом вышеописанных соглашений, то взаимное исключение гарантируется. При этом потоки могут быть прерваны операционной системой в любой момент и в любом месте, в том числе в критической секции.

Однако следует заметить, что одно ограничение на прерывания все же имеется. Нельзя прерывать поток между выполнением операций проверки и установки блокирующей переменной. Поясним это. Пусть в результате проверки переменной поток определил, что ресурс свободен, но сразу после того, не успев установить переменную в 0, был прерван. За время его приостановки другой поток занял ресурс, вошел в свою критическую секцию, но также был прерван, не завершив работы с разделяемым ресурсом. Когда управление было возвращено первому потоку, он, считая ресурс свободным, установил признак занятости и начал выполнять свою критическую секцию. Таким образом, был нарушен принцип взаимного исключения, что потенциально может привести к нежелательным последствиям. Во избежание таких ситуаций в системе команд многих компьютеров предусмотрена единая,

неделимая команда анализа и присвоения значения логической переменной. При отсутствии такой команды в процессоре соответствующие действия должны реализовываться специальными системными примитивами, которые бы запрещали прерывания на протяжении всей операции проверки и установки.

Реализация взаимного исключения описанным выше способом имеет существенный недостаток: в течение времени, когда один поток находится в критической секции, другой поток, которому требуется тот же ресурс, получив доступ к процессору, будет непрерывно опрашивать блокирующую переменную, бесполезно тратя выделяемое ему процессорное время, которое могло бы быть использовано для выполнения какого-нибудь другого потока. Для устранения этого недостатка во многих ОС предусматриваются специальные системные вызовы для работы с критическими секциями.

Раздел 3 Ресурсы в составе операционных систем

Тема 3.1 Драйверы устройств

Одной из главных задач операционной системы является обеспечение обмена данными между приложениями и периферийными устройствами компьютера. В современной ОС функции обмена данными с периферийными устройствами выполняет подсистема ввода-вывода. Клиентами этой подсистемы являются не только пользователи и приложения, но и некоторые компоненты самой ОС, которым требуется получение системных данных или их вывод, например, подсистеме управления процессами при смене активного процесса необходимо записать на диск контекст приостанавливаемого процесса и считать с диска контекст активизируемого процесса.

Основными компонентами подсистемы ввода-вывода являются драйверы, управляющие внешними устройствами, и файловая система. Файловая система как основное хранилище всей информации вычислительной системы рассматривается совместно с другими компонентами подсистемы ввода-вывода по двум причинам. Во-первых, файловая система активно использует остальные части подсистемы ввода-вывода, а во-вторых, модель файла лежит в основе большинства механизмов доступа к устройствам, используемых в современной подсистеме ввода-вывода.

Подсистема ввода-вывода (Input-Output Subsystem) мультипрограммной ОС при обмене данными с внешними устройствами компьютера должна решать ряд общих задач, из которых наиболее важными являются следующие:

- организация параллельной работы устройств ввода-вывода и процессора;
- согласование скоростей обмена и кэширование данных;
- разделение устройств и данных между процессами;
- обеспечение удобного логического интерфейса между устройствами и остальной частью системы;
- поддержка широкого спектра драйверов с возможностью простого включения в систему нового драйвера;
- динамическая загрузка и выгрузка драйверов;
- поддержка нескольких файловых систем;
- поддержка синхронных и асинхронных операций ввода-вывода.

Рассмотрим перечисленные задачи более подробно.

Организация параллельной работы устройств ввода-вывода и процессора. Каждое устройство ввода-вывода вычислительной системы диск, принтер, терминал и т. п. снабжено специализированным блоком управления, называемым контроллером. Контроллер взаимодействует с драйвером системным программным модулем, предназначенным для управления данным устройством. Контроллер периодически принимает от драйвера выводимую на устройство информацию, а также команды управления, которые говорят о том, что с этой информацией нужно сделать (например, вывести в виде текста в определенную область терминала или записать в определенный сектор диска). Под управлением контроллера

устройство может некоторое время выполнять свои операции автономно, не требуя внимания со стороны центрального процессора.

Процессы, происходящие в контроллерах, протекают в периоды между выдачами команд независимо от ОС. От подсистемы ввода-вывода требуется спланировать в реальном масштабе времени (в котором работают внешние устройства) запуск и приостановку большого количества разнообразных драйверов, обеспечив приемлемое время реакции каждого драйвера на независимые события контроллера. При этом необходимо минимизировать загрузку процессора задачами ввода-вывода, оставив как можно больше процессорного времени на выполнение пользовательских потоков.

Согласование скоростей обмена и кэширование данных. При обмене данными всегда возникает задача согласования скорости. Например, если один пользовательский процесс вырабатывает некоторые данные и передает их другому пользовательскому процессу через оперативную память, то в общем случае скорости генерации данных и их чтения не совпадают. Согласование скорости обычно достигается за счет буферизации данных в оперативной памяти и синхронизации доступа процессов к буферу.

В подсистеме ввода-вывода для согласования скоростей обмена также широко используется буферизация данных в оперативной памяти. В тех специализированных операционных системах, в которых обеспечение высокой скорости ввода-вывода является первоочередной задачей (управление в реальном времени, услуги сетевой файловой службы и т. п.), большая часть оперативной памяти отводится не под коды прикладных программ, а под буферизацию данных. Однако буферизация только на основе оперативной памяти в подсистеме ввода-вывода оказывается недостаточной. Разница между скоростью обмена с оперативной памятью, куда процессы помещают данные для обработки, и скоростью работы внешнего устройства часто становится слишком значительной, чтобы в качестве временного буфера можно было бы использовать оперативную память ее объема может просто не хватить. Для таких случаев необходимо предусмотреть особые меры, и часто в качестве буфера используется дисковый файл, называемый также спул-файлом (от spool шпулька, тоже буфер, только для ниток). Типичный пример применения спулинга дает организация вывода данных на принтер.

Другим решением этой проблемы является использование большой буферной памяти в контроллерах внешних устройств. Такой подход особенно полезен в тех случаях, когда помещение данных на диск слишком замедляет обмен (или когда данные выводятся на сам диск). Например, в контроллерах графических дисплеев применяется буферная память, соизмеримая по объему с оперативной, и это существенно ускоряет вывод графики на экран.

Буферизация данных позволяет не только согласовать скорости работы процессора и внешнего устройства, но и решить другую задачу – сократить количество реальных операций ввода-вывода за счет кэширования данных. Дисковый кэш является непременным атрибутом подсистем ввода-вывода практически всех операционных систем, значительно сокращая время доступа к хранимым данным.

Поддержка широкого спектра драйверов и простота включения нового драйвера в систему.

Достоинством подсистемы ввода-вывода любой универсальной ОС является наличие разнообразного набора драйверов для наиболее популярных периферийных устройств. Прекрасно спланированная и реализованная операционная система может потерпеть неудачу на рынке только из-за того, что в ее состав не включен достаточный набор драйверов. В этом случае администраторы и пользователи вынуждены искать нужный им драйвер для имеющегося у них внешнего устройства у производителей оборудования или, что еще хуже, заниматься его разработкой самостоятельно.

Драйвер взаимодействует, с одной стороны, с модулями ядра ОС (модулями подсистемы ввода-вывода, модулями системных вызовов, модулями подсистем управления процессами и памятью и т. д.), а с другой стороны с контроллерами внешних устройств. Поэтому существуют два типа интерфейсов: интерфейс «драйвер-ядро» (Driver Kernel

Interface, DKI) и интерфейс «драйвер-устройство» (Driver Device Interface, DDI). Интерфейс «драйвер-ядро» должен быть стандартизован в любом случае. Интерфейс «драйвер-устройство» имеет смысл стандартизировать тогда, когда подсистема ввода-вывода не разрешает драйверу непосредственно взаимодействовать с аппаратурой контроллера и выполняет эти операции самостоятельно. Экранирование драйвера от аппаратуры является весьма полезной функцией, так как драйвер в этом случае становится независимым от аппаратной платформы. Подсистема ввода-вывода может поддерживать несколько различных типов интерфейсов DKI/DDI, предоставляя специфический интерфейс для устройств определенного класса.

Для поддержки процесса разработки драйверов операционной системы обычно выпускается так называемый пакет DDK (Driver Development Kit), представляющий собой набор соответствующих инструментальных средств библиотек, компиляторов и отладчиков.

Динамическая загрузка и выгрузка драйверов. Кроме проблемы разработки новых драйверов существует также проблема включения драйвера в состав модулей работающей ОС, то есть динамической загрузки-выгрузки драйвера. Так как набор потенциально поддерживаемых данной ОС периферийных устройств всегда существенно шире набора устройств, которыми ОС должна управлять при установке на конкретной машине, то ценным свойством ОС является возможность динамически загружать в оперативную память требуемый драйвер (без останова ОС) и выгружать его после того, как потребность в поддержке устройства миновала, что может существенно сэкономить системную область памяти.

Альтернативой динамической загрузке драйверов при изменении текущей конфигурации внешних устройств компьютера является повторная компиляция кода ядра с требуемым набором драйверов, что создает между всеми компонентами ядра статические связи вместо динамических. Например, таким образом решалась данная проблема в ранних версиях операционной системы UNIX. При статических связях между ядром и драйверами структура ОС упрощается, но этот подход требует наличия исходных кодов модулей операционной системы, доступность которых скорее является исключением (для некоммерческих версий UNIX), а не правилом. Кроме того, в этом варианте работающую предыдущую версию операционной системы необходимо остановить и заменить новой, а перерывы в работе ОС в некоторых применениях могут и не допускаться. Поддержка динамической загрузки драйверов является практически обязательным требованием для современных универсальных операционных систем.

Поддержка нескольких файловых систем. Диски представляют особый род периферийных устройств, так как именно на них хранится большая часть как пользовательских, так и системных данных. Данные на дисках организуются в файловые системы, и свойства файловой системы во многом определяют свойства самой ОС ее отказоустойчивость, быстродействие, максимальный объем хранимых данных. Популярность файловой системы часто приводит к ее миграции из «родной» ОС в другие операционные системы. Например, файловая система FAT появилась первоначально в MS-DOS, но затем была реализована в OS/2, семействе MS Windows и многих реализациях UNIX. Ввиду этого поддержка нескольких популярных файловых систем для подсистемы ввода-вывода также важна, как и поддержка широкого спектра периферийных устройств. Важно также, чтобы архитектура подсистемы ввода-вывода позволяла достаточно просто включать в ее состав новые типы файловых систем, без необходимости переписывания кода.

Тема 3.2 Настройка параметров работы устройств для взаимодействия с операционной системой

Способы управления и настройки устройств вычислительной системы. Организация взаимодействия между устройствами. Назначение и структура BIOS. Ресурсы BIOS. Функциональная характеристика устройства. Процедуры инициализации и POST-

диагностики. Цифровая и звуковая индикация ошибок. Передача управления загрузчику операционной системы. Перепрошивка BIOS. Моддинг BIOS. Программа CMOS. Основная страница конфигурации. Расширенные страницы конфигурации. Оверклокинг. Защита CMOS на уровне пароля. Защита настроек. Ограничение доступа. Шифрование оборудования. Настройка устройств средствами современных операционных систем.

Тема 3.3 Управление памятью

Под памятью (memory) здесь подразумевается оперативная память компьютера. В отличие от памяти жесткого диска, которую называют внешней памятью (storage), оперативной памяти для сохранения информации требуется постоянное электропитание.

Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы. Особая роль памяти объясняется тем, что процессор может выполнять инструкции программы только в том случае, если они находятся в памяти. Память распределяется как между модулями прикладных программ, так и между модулями самой операционной системы.

Помимо первоначального выделения памяти процессам при их создании ОС должна также заниматься динамическим распределением памяти, то есть выполнять запросы приложений на выделение им дополнительной памяти во время выполнения. После того как приложение перестает нуждаться в дополнительной памяти, оно может возвратить ее системе. Выделение памяти случайной длины в случайные моменты времени из общего пула памяти приводит к фрагментации и, вследствие этого, к неэффективному ее использованию. Дефрагментация памяти тоже является функцией операционной системы.

Во время работы операционной системы ей часто приходится создавать новые служебные информационные структуры, такие, как описатели процессов и потоков, различные таблицы распределения ресурсов, буферы, используемые процессами для обмена данными, синхронизирующие объекты и т. п. Все эти системные объекты требуют памяти. В некоторых ОС заранее (во время установки) резервируется некоторый фиксированный объем памяти для системных нужд. В других же ОС используется более гибкий подход, при котором память для системных целей выделяется динамически. Таким образом, разные подсистемы ОС при создании своих таблиц, объектов, структур и т. п. также обращаются к подсистеме управления памятью с запросами.

Защита памяти - это еще одна важная задача операционной системы. Эта функция, как правило, реализуется программными модулями ОС в тесном взаимодействии с аппаратными средствами.

Все алгоритмы разделены на два класса: алгоритмы, в которых используется перемещение сегментов процессов между оперативной памятью и диском, и алгоритмы, в которых внешняя память не привлекается.

Память вычислительной машины представляет собой иерархию запоминающих устройств (ЗУ), отличающихся средним временем доступа к данным, объемом и стоимостью хранения одного бита. Фундаментом этой пирамиды запоминающих устройств служит внешняя память, как правило, представляемая жестким диском. Она имеет большой объем (десятки и сотни гигабайт), но скорость доступа к данным является невысокой.

Тема 3.4 Понятие виртуального ресурса

Концентрация виртуального ресурса. Конфигурация виртуальной машины включает в себя виртуальный процессор; виртуальную память (основную память); виртуальные каналы ввода-вывода; виртуальные устройства ввода-вывода; виртуальный пульт оператора.

Виртуальный процессор (виртуальный центральный процессор) моделируется на реальном процессоре путем выделения каждой ВМ части времени реального процессора по принципу квантования.

Виртуальная память (ВП) моделируется на реальной оперативной памяти с использованием средства динамического преобразования адресов. Адресное пространство

виртуальной памяти может превышать адресное пространство реальной памяти. Оно ограничивается максимально допустимым размером адреса в системе. С точки зрения операционной системы, выполняемой в виртуальной машине, виртуальная память рассматривается как основная. Операционная система может организовать собственную виртуальную память. Концепция виртуальной памяти используется не только в рамках концепции виртуальной машины, но часто и самостоятельно в операционных системах (операционные системы с виртуальной памятью).

Виртуальные каналы ввода-вывода моделируются либо путем разделения реальных каналов ввода-вывода, либо путем закрепления (распределения) реального канала за виртуальной машиной. Аналогично моделируются и устройства управления. Способ моделирования зависит от того, как моделируются виртуальные устройства ввода-вывода.

Виртуальные устройства ввода-вывода моделируются путем разделения, накопления или закрепления.

Разделение используется для моделирования накопителей на магнитных дисках (НМД). Разделение означает одновременное использование реального НМД несколькими виртуальными машинами. Путем разделения одного реального НМД моделируется несколько виртуальных магнитных дисков того же типа. При этом реальный том разбивается на участки, называемые мини-дисками, содержащими целое число смежных цилиндров.

Накопление используется для обеспечения виртуальных машин низкоскоростными устройствами последовательного доступа. Накопление моделирует реальное устройство в области накопления, выделяемой на НМД. Входные файлы предварительно считываются МВМ в область накопления с реальных устройств. Затем область накопления используется для моделирования виртуальных устройств. Выходные файлы, предназначенные для устройств вывода, предварительно записываются в область накопления, а затем МВМ осуществляет их реальный вывод. Допускается считывание выходных файлов непосредственно из области накопления без вывода на реальное устройство.

Закрепление (распределение) означает выделение реального устройства в полное распоряжение виртуальной машины. Может применяться для любого устройства, однако применяется в основном для накопителей на магнитных лентах (НМЛ), абонентских пунктов и других устройств, не обеспеченных в СВМ.

Виртуальный пульт оператора использует либо пультовое устройство реальной ЭВМ, либо абонентский пункт (локальный или удаленный). Посредством виртуального пульта оператора выполняются функции, аналогичные функциям пульта реальной ЭВМ, а также осуществляется связь с операционной системой в виртуальной машине.

Тема 3.5 Создание и управление виртуальными машинами

Виды гипервизоров. Гипервизоры типа 1 и типа 2. Разделяемы ресурсы виртуальных машин. Сетевое взаимодействие виртуальных машин. Гипервизор с открытым кодом KVM. Hyper-V. VMware. ESXi. Vagrant. Docker. Кластеры виртуализации. Балансировка нагрузки.

Обобщение понятия виртуального ресурса приводит к концепции виртуальной машины. Виртуальная машина (ВМ) является функциональным эквивалентом реальной ЭВМ, который моделируется при помощи реальных технических средств и совокупности управляющих программ, называемых монитором виртуальных машин (МВМ). Конфигурация виртуальной машины отличается от конфигурации реальной ЭВМ. На одной реальной ЭВМ может функционировать несколько виртуальных машин. Программное обеспечение виртуальных машин, основой которого является МВМ, представляет собой систему виртуальных машин (СВМ).

Система виртуальных машин обеспечивает новый принцип параллелизма (мультипрограммирования) в работе реальной ЭВМ. Каждому пользователю предоставляется отдельная ВМ, в которой может функционировать:

— версия операционной системы;

— собственная системно-независимая программа;

— подсистема СВМ,, например подсистема диалоговой обработки (ПДО);

— подсистема операционной системы, основанная на концепции ВМ.

Пользователи отдельных ВМ полностью независимы.

Система виртуальных машин решает проблемы «традиционных» операционных систем, упомянутые ранее.

1. Введение концепции виртуальной машины позволяет уменьшить взаимное влияние различных режимов использования вычислительной системы. Упрощается управляющая программа.

2. Пользователь имеет иллюзию обладания отдельной ЭВМ. В действительности же он обладает отдельной виртуальной машиной. Управляющая программа в большей степени «прозрачна».

3. Виртуальная основная память позволяет снизить остроту проблемы «фрагментации» памяти.

4. Уровень реального мультипрограммирования повышается. В результате возрастает интенсивность использования ресурсов, в первую очередь ЦП и оперативной памяти.

5. Виртуализация расширяет объем ресурсов, в первую очередь основной памяти.

6. На одной реальной ЭВМ можно выполнить одновременно несколько операционных систем, их версий, подсистем, системно-независимых программ.

Раздел 4 Гетерогенное взаимодействие операционных систем

Тема 4.1 Операционные системы сетевых узлов

Понятие гетерогенности. Понятие мультиплатформенности. Модель сетевого взаимодействия ISO OSI. Протоколы сетевого взаимодействия. Пиринговое взаимодействие (p2p). Взаимодействие «клиент-сервер». Операционные системы клиентов. Веб-интерфейс и консольное управление. Операционные системы серверов. Операционные системы коммутирующих устройств. Операционные системы маршрутизаторов. Сбор статистик. Фильтрация сетевого трафика. Регистрация ошибок. Механизмы информирования.

Тема 4.2 Операционные системы IoT

Функционал операционных систем устройств IoT. Требования к операционным системам IoT: минимальное потребление памяти; энергоэффективность; возможности подключения к сети; аппаратно-независимые операции; требования к работе в реальном времени; требования безопасности; экосистема разработки приложений. Концепция «туманных» вычислений. API операционных систем устройств IoT. Примеры операционных системы IoT: Ubuntu Core; RIOT; Contiki; TinyOS; Zephyr.

Тема 4.3 Процедуры обеспечения безопасности

Необходимые компоненты базовой локальной политики безопасности. Задачи по защите оборудования. Способы защиты данных. Методы обеспечения безопасности в беспроводных сетях. Распространенные профилактические меры по обеспечению безопасности. Обновление файлов подписей для программ защиты от вирусов и шпионского ПО. Установка пакетов обновлений и исправлений для операционных систем. Устранение проблем в обеспечении безопасности. Обзор процедуры устранения проблем. Определение распространенных проблем и решений.

Безопасность данных вычислительной системы обеспечивается средствами отказоустойчивости ОС, направленными на защиту от сбоев и отказов аппаратуры и ошибок программного обеспечения, а также средствами защиты от несанкционированного доступа. В последнем случае ОС защищает данные от ошибочного или злонамеренного поведения пользователей системы. Первым рубежом обороны при защите данных от несанкционированного доступа является процедура логического входа. Операционная система должна убедиться, что в систему пытается войти пользователь, вход которого

разрешен администратором. Функции защиты ОС вообще очень тесно связаны с функциями администрирования, так как именно администратор определяет права пользователей при их обращении к разным ресурсам системы - файлам, каталогам, принтерам, сканерам и т. п. Кроме того, администратор ограничивает возможности пользователей в выполнении тех или иных системных действий. Например, пользователю может быть запрещено выполнять процедуру завершения работы ОС, устанавливать системное время, завершать чужие процессы, создавать учетные записи пользователей, изменять права доступа к некоторым каталогам и файлам. Администратор может также урезать возможности пользовательского интерфейса, убрав, например, некоторые пункты из меню операционной системы, выводимого на дисплей пользователя.

Важным средством защиты данных являются функции аудита ОС, заключающиеся в фиксации всех событий, от которых зависит безопасность системы. Например, попытки удачного и неудачного логического входа в систему, операции доступа к некоторым каталогам и файлам, использование принтеров и т. п. Список событий, которые необходимо отслеживать, определяет администратор ОС.

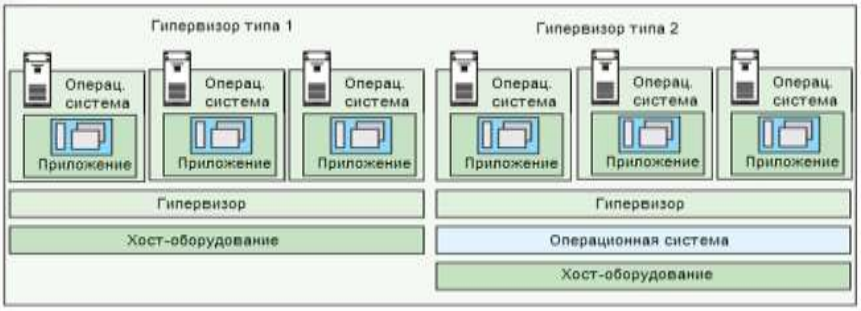
Поддержка отказоустойчивости реализуется операционной системой, как правило, на основе резервирования. Чаще всего в функции ОС входит поддержание нескольких копий данных на разных дисках или разных дисковых накопителях. Резервируются также принтеры и другие устройства ввода-вывода. При отказе одного из избыточных устройств операционная система должна быстро и прозрачным для пользователя образом произвести реконфигурацию системы и продолжить работу с резервным устройством. Поддержка отказоустойчивости также входит в обязанности системного администратора. В состав ОС обычно входят утилиты, позволяющие администратору выполнять регулярные операции резервного копирования для обеспечения быстрого восстановления важных данных.

Тема 4.4 Построение изолированных программных сред

Аппаратно изолированная среда. Использование защищенных сетевых структур в публичном облаке. Сетевые политики. DMZ. Доверительные отношения между сетевыми сегментами. Тоннели. Защищенные тоннели. Виртуализация узлов. Программно-изолированная среды при межпроцессном управлении операционной системы. Контейнерный подход. Понятие «песочницы»

Пример презентационного материала для проведения занятия

Существует два типа гипервизоров:



Гипервизоры поддерживают различные аппаратные платформы в различных облачных средах. Например:

PowerVM: принадлежность серверов на базе IBM POWER5, POWER6 и POWER7, этот гипервизор поддерживается операционными системами IBM i, AIX® и Linux®; PowerVM поддерживается в среде IBM SmartCloud Enterprise.

VMware ESX Server: встроенный гипервизор VMware ESX работает непосредственно на аппаратуре серверов, не требуя дополнительной операционной системы. Он поддерживается в среде IBM SmartCloud Enterprise.

Xen: монитор виртуальных машин для процессорных архитектур IA-32, x86-64, Itanium и ARM, Xen позволяет выполнять несколько гостевых операционных систем на одном и том же оборудовании одновременно. Xen-системы имеют структуру, в которой гипервизор Xen занимает самый низкий и привилегированный уровень.

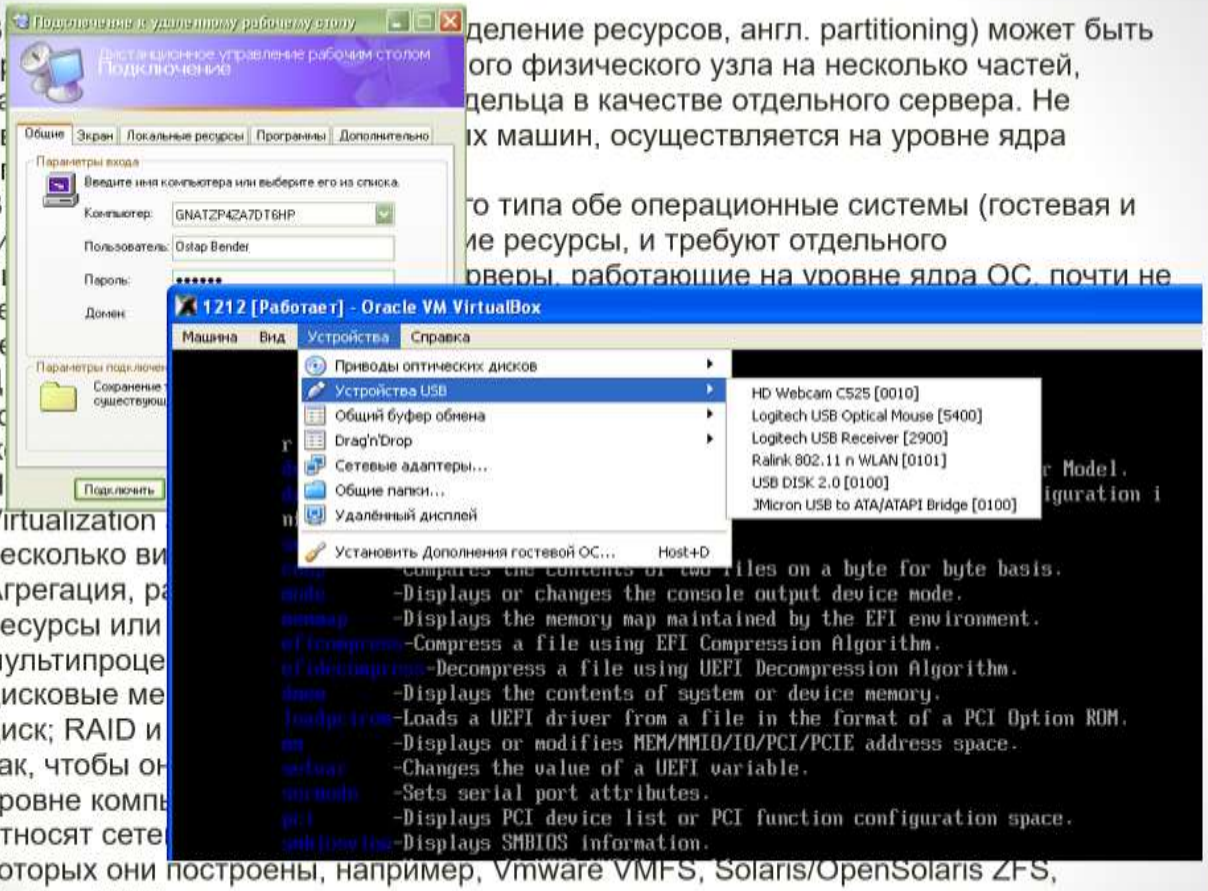
KVM: инфраструктура виртуализации для ядра Linux, KVM поддерживает платформенно-зависимую виртуализацию на процессорах с аппаратными расширениями для виртуализации. Первоначально он поддерживал процессоры x86, но в настоящее время к ним добавился широкий спектр процессоров и гостевых операционных систем, в том числе множество вариаций Linux, BSD, Solaris, Windows®, Haiku, **ReactOS** и **AROS Research Operating System** (есть даже модифицированная версия QEMU, способная использовать KVM для работы с Mac OS X).

z/VM: текущая версия операционной системы виртуальных машин IBM, z/VM работает на серверах IBM zSeries и может использоваться для поддержки большого числа (тысяч) виртуальных машин Linux..

Эта модель использует менеджер виртуальных машин (гипервизор), который осуществляет связь между гостевой операционной системой и аппаратными средствами системы.

Внутри гипервизора должна быть установлена и настроена определенная защита, потому, что основные аппаратные средства не принадлежат ОС, а разделяются гипервизором. При построении крупных корпоративных систем, как правило, используется именно аппаратная виртуализация. При этом крупные вендоры такие как VMware, IBM и Microsoft разрабатывают свои платформы виртуализации на базе технологий аппаратной виртуализации Intel VT (VT-x), AMD-V.

В деление ресурсов, англ. partitioning) может быть
пр ого физического узла на несколько частей,
ка дельца в качестве отдельного сервера. Не
яв их машин, осуществляется на уровне ядра
ог
В го типа обе операционные системы (гостевая и
ги ие ресурсы, и требуют отдельного
ли рверы, работающие на уровне ядра ОС, почти не
те
се
Д
ко
ж
Н
Virtualization
несколько ви
Агрегация, ра
ресурсы или
мультипроце
дисковые ме
диск; RAID и
так, чтобы он
уровне компь
относят сете
которых они построены, например, Vmware VMFS, Solaris/OpenSolaris ZFS, NetApp WAFL.



Виды устройств: домашние роутеры

В качестве примера остановимся на двухдиапазонном Wi-Fi маршрутизаторе ASUS RT-AC86U

Происхождение
- Основы концепции заложил в 1966 Дональд Дэвис для британской сети NPL

Сфера применения
- изначально разработан как устройство подключения к сети провайдера Small office/home office (SOHO)

Объем ресурсов
- Процессор: 2х-ядерный ARM-процессор с частотой 1,8 ГГц
- Память: RAM объемом 512 Мб
- Wi-Fi: 2.4GHz/5GHz, 4xEthernet 1 Гбит/с, WAN
- технология **AiProtection** позволяет обеспечить защиту всех имеющихся подключений к роутеру, которые при этом могут не обладать собственными антивирусными системами.

Операционная система
- Проприетарная

https://www.ixbt.com/nw/asus-rt-ac86u-review.html#n4

Динамика роста диагоналей смартфонов, 2015-2019 гг. (в шт.)

- изначал
  как сред

Объем ре

- Процесс
  (1x2.95
- Память:
- Видеопр
- Аккумул

Операци

- Google
- Apple iO
- ...

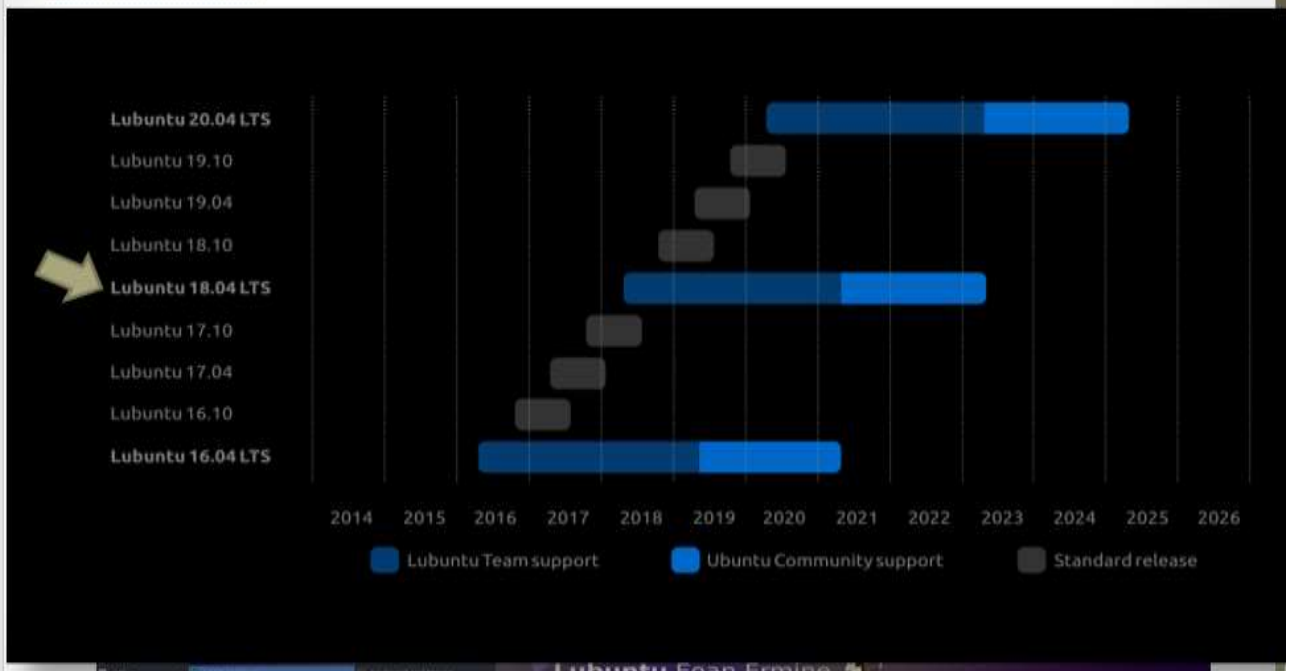
Linux Mint

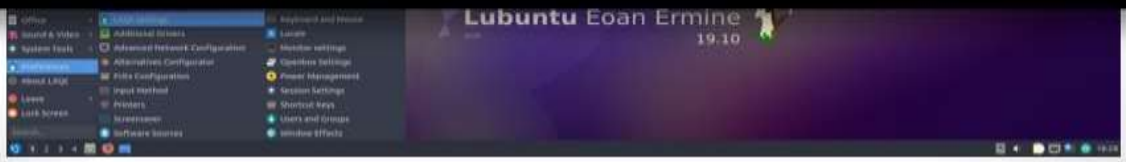
Это лучший вариант для тех, кто никогда не имел дела с Linux. Mint очень прост в освоении, стабильно работает даже на самом старом железе и симпатично выглядит. У него есть вариация с оболочкой Cinnamon для более современных устройств и Xfce для старых машин.

Система предоставляет удобный магазин приложений (никакой возни с «Терминалом») и наглядное меню настроек. Mint куда быстрее, чем Windows 7, не говоря уже о «Десятке».

Минимальные системные требования: процессор 1 ГГц, 1 ГБ оперативной памяти, 15 ГБ свободного места на жёстком диске.

Lubuntu





HTML/ CSS/ JavaScript 1500 ˅ 300 200 / 8000 1500

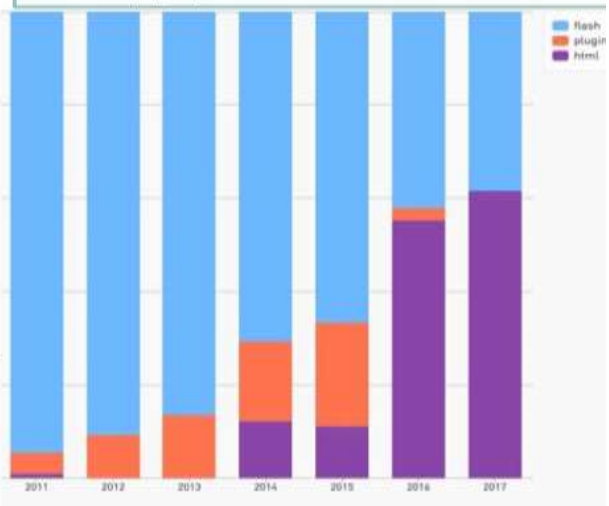
Язык для структурирования и представления содержимого всемирной паутины.

Хотя стандарт был завершён только в 2014 году (предыдущая, четвёртая, версия опубликована в 1999 году), уже с 2013 года браузерами оперативно осуществлялась поддержка, а разработчиками — использование рабочего стандарта (HTML Living Standard).

Цель разработки HTML5 — улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров.

Ведет историю с 1992 года.

Если взять только «качественные» игры
оценкой больше или равной 4 и больше
игровых сессий, то процент HTML5 игр
2017 году превысил 60%



3 ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

Назначение операционных систем

1. Эволюция операционных систем.
2. Классификация ОС.
3. Системы виртуальных машин.

Интерфейсы операционных систем

4. Пакетный интерфейс.
5. Графический интерфейс.
6. Речевой интерфейс и интерфейс распознавания образов.
7. Тактильный интерфейс.
8. Семантический интерфейс.

Функции операционных систем

9. Функции ОС по управлению ресурсами.
10. Управление памятью.
11. Управление файлами и внешними устройствами.

Процессы и потоки

12. Понятия «процесс» и «поток».
13. Создание процессов и потоков.
14. Контекст и дескриптор процесса.

Алгоритмы планирования

15. Циклическое планирование.
16. Многоуровневые очереди с обратными связями.
17. Квантование с предпочтением потоков.

Синхронизация процессов и потоков

18. Необходимость синхронизации и гонки.
19. Понятие критической секции.
20. Задачи синхронизации.

Тупиковые ситуации

21. Тупиковые ситуации.
22. Подходы к решению тупиковых ситуаций.
23. Алгоритмы обнаружения тупиков.

Драйверы устройств

24. Понятие драйвера устройства.
25. Установка драйверов в различных операционных системах.
26. Порядок использования драйверов в системе.

Настройка работы устройств для взаимодействия с операционной системой

27. Способы управления и настройки устройств в вычислительных системах.
28. Назначение и структура BIOS.
29. Программа CMOS.
30. Настройка устройств средствами современных операционных систем.

Управление памятью

31. Функции ОС по управлению памятью.
32. Методы распределения памяти.
33. Иерархия запоминающих устройств.

Понятие виртуального ресурса

34. Организация виртуального ресурса.
35. Виртуальный процессор.
36. Виртуальная память.

Создание и управление виртуальными машинами

37. Гипервизоры типа 1 и типа 2.
38. Разделяемы ресурсы виртуальных машин.
39. Сетевое взаимодействие виртуальных машин.

Операционные системы сетевых узлов

40. Веб-интерфейс и консольное управление.
41. Фильтрация сетевого трафика.
42. Протоколы сетевого взаимодействия.

Операционные системы IoT

43. Функционал операционных систем устройств IoT.
44. Концепция «туманных» вычислений.
45. Энергообеспечение устройств IoT.
46. API операционных систем устройств IoT.

Процедуры обеспечения безопасности

47. Задачи по защите оборудования на уровне операционных систем.
48. Способы защиты данных.
49. Профилактические меры по обеспечению безопасности.

Построение изолированных программных сред

50. Аппаратно изолированная среда.
51. Использование защищенных сетевых структур в публичном облаке.
52. Контейнерный подход.

4 ЗАДАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

Лабораторная работа №1 Синхронизация процессов и потоков

Задание: Регистрация процессов в системе и обнаружение своего окружения. Организация системы связи между процессами. Управление процессами и потоками.

Лабораторная работа №2 Настройка параметров работы устройств для взаимодействия с операционной системой

Задание: Получение практических навыков по контролю состава оборудования, программных средств и настройки операционных систем различных производителей.

Лабораторная работа №3 Создание и управление виртуальными машинами

Задание: Создание виртуального стенда взаимодействующих операционных систем в программной среде гипервизора 2 типа.

Лабораторная работа №4 Процедуры обеспечения безопасности

Задание: Изучение основы защиты информации в вычислительных сисстемах и компьютерных сетях. Получение навыков по управление доступом, идентификации и установление подлинности, работе со средствами идентификации и аутентификации.

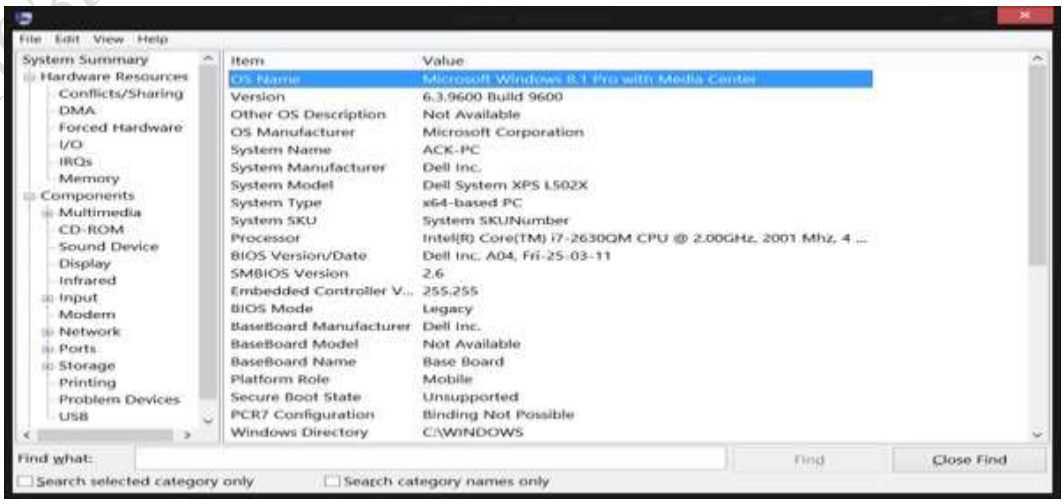
Примеры реализации лабораторных работ:

Лабораторная работа №2 Настройка параметров работы устройств для взаимодействия с операционной системой

Часть 1. Сбор данных о драйверах Операционной системы

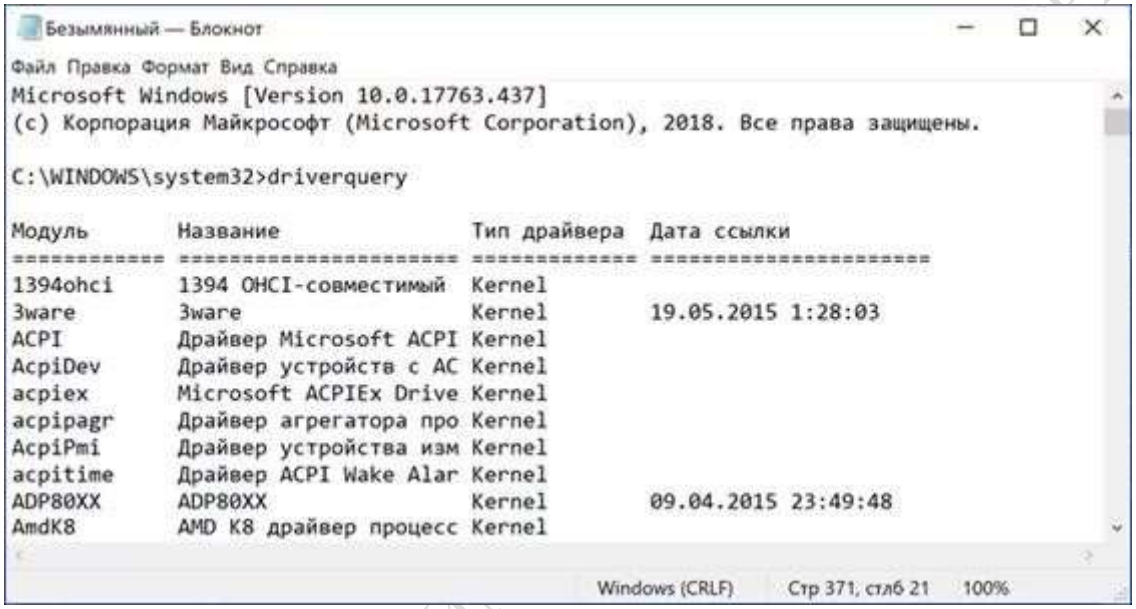
В общем виде драйвер представляет собой программную прослойку между операционной системой и аппаратной конфигурацией компьютера. В его задачу входит перевод поступающих от ОС команд на понятный для устройств язык и установка между ними обратной связи.

Инструмент Windows «Сведения о системе» (утилита msinfo32) входит в состав операционных систем Microsoft еще с версии Windows 98. С ее помощью можно получить практически полные данные о компьютере, включая и перечень установленных драйверов. Вызываем клавиатурным сочетанием «Win+R» окно «Выполнить». Набираем в текстовом поле «msinfo32». Запускаем исполнение введенной команды нажав «ОК» или клавишу ввода.



1. Просмотр драйверов на Windows в CMD

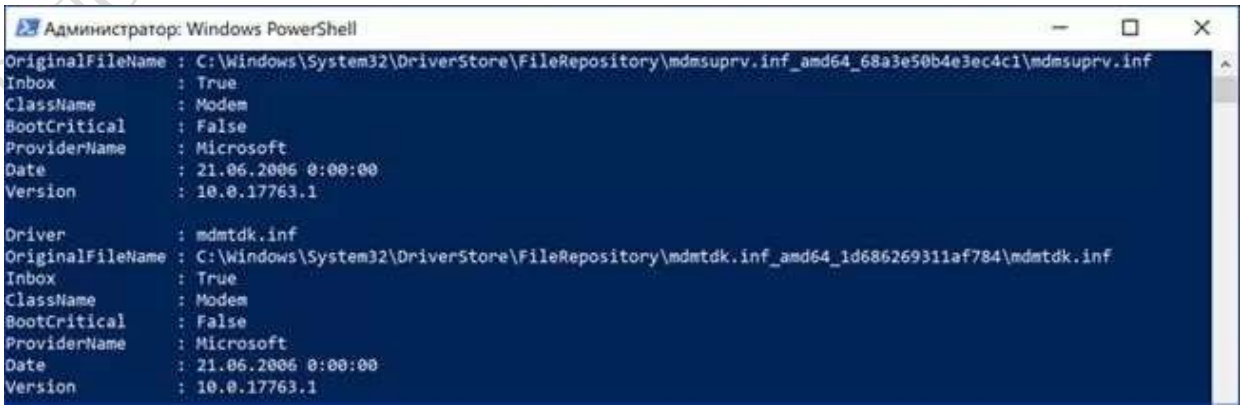
Весь перечень установленных в Windows драйверов, можно получить с помощью командной строки. В дополнительном меню кнопки «Пуск» используем отмеченный пункт, чтобы запустить консоль с административными привилегиями. Набираем команду «driverquery». В окне выводится полный список всех драйверов, отсортированный в алфавитном порядке. Для удобства изучения его можно скопировать с помощью горячих клавиш. Выделяем все содержимое окна командной строки сочетанием «Ctrl+A». Копируем информацию нажимая «Ctrl+C», и вставляем ее в Блокнот комбинацией «Ctrl+V». Применение дополнительных ключей позволяет выводить детализированную информацию. Для примера показана команда «driverquery /si». С ее помощью открывается список только подписанных драйверов. Полный перечень дополнительных параметров можно получить, набрав «driverquery /?».



2. Просмотр драйверов на Windows в PowerShell

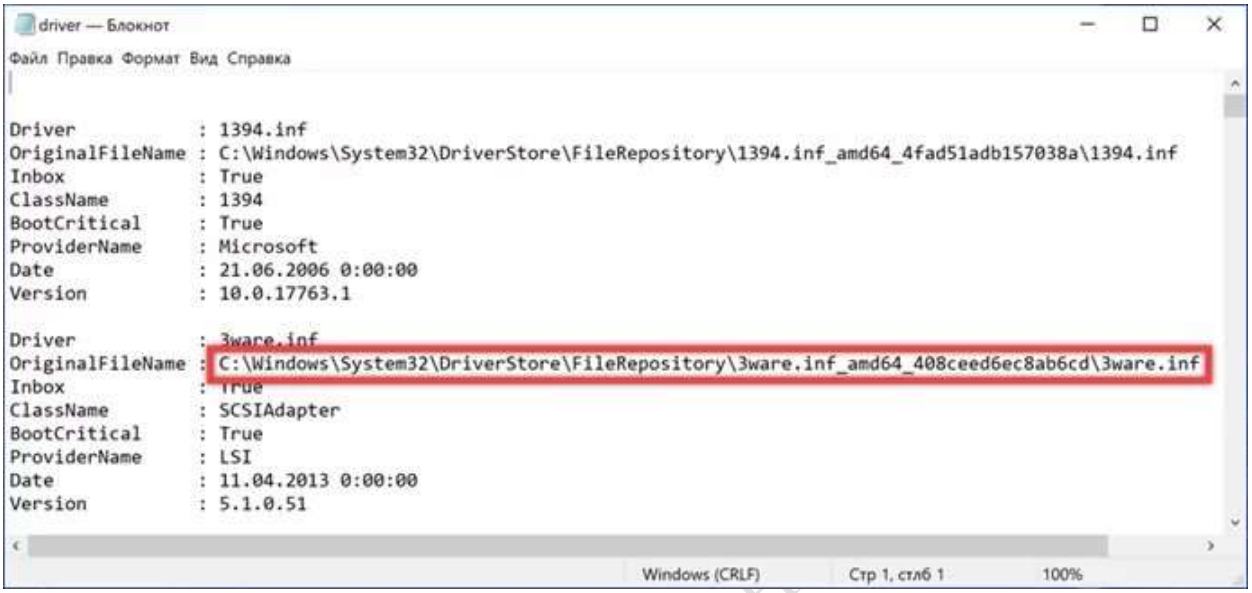
Оболочка PowerShell разработана Microsoft и продвигается в качестве современной альтернативы консоли CMD. Кроме поддержки старых команд в ней используется объектно-ориентированные сценарии. Открываем дополнительное меню кнопки «Пуск». Запускаем PowerShell в режиме повышенных привилегий.

Вводим команду «Get-WindowsDriver -online -all». Ключ «all» предназначен для вывода данных о всех драйверах, установленных на локальном компьютере. Если его не использовать в список включается только программное обеспечение сторонних производителей. Обработка введенной команды представляется в виде, показанном на скриншоте.



Массив полученных данных не вмещается в заложенные по умолчанию 3000 строк. Чтобы изучать его с экрана размер вывода нужно заранее увеличить в настройках Powershell либо перенаправить вывод команды в файл на внешнем носителе.

Пример синтаксиса: «Get-WindowsDriver -online -all | Out-File driver.txt». Итоговый документ driver.txt сохраняется в системном каталоге.

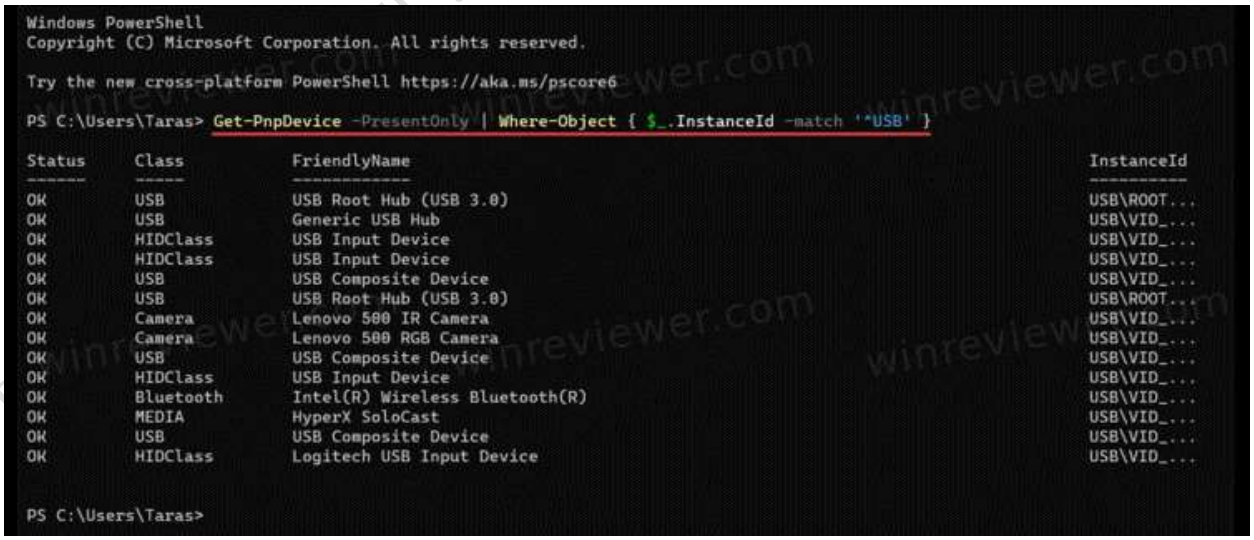


3. Получить список подключенных USB-устройств в Windows

Введите следующую команду:

«Get-PnpDevice -PresentOnly | Where-Object { $_. InstanceId -match '^USB'}»

Эта команда покажет список всех имеющихся USB-устройств. «Статус ОК» означает, что устройство в настоящее время подключено и работает правильно. Вы также можете использовать столбцы Class и Friendly Name, чтобы быстро найти и лучше распознать устройства, которые вы видите в списке.

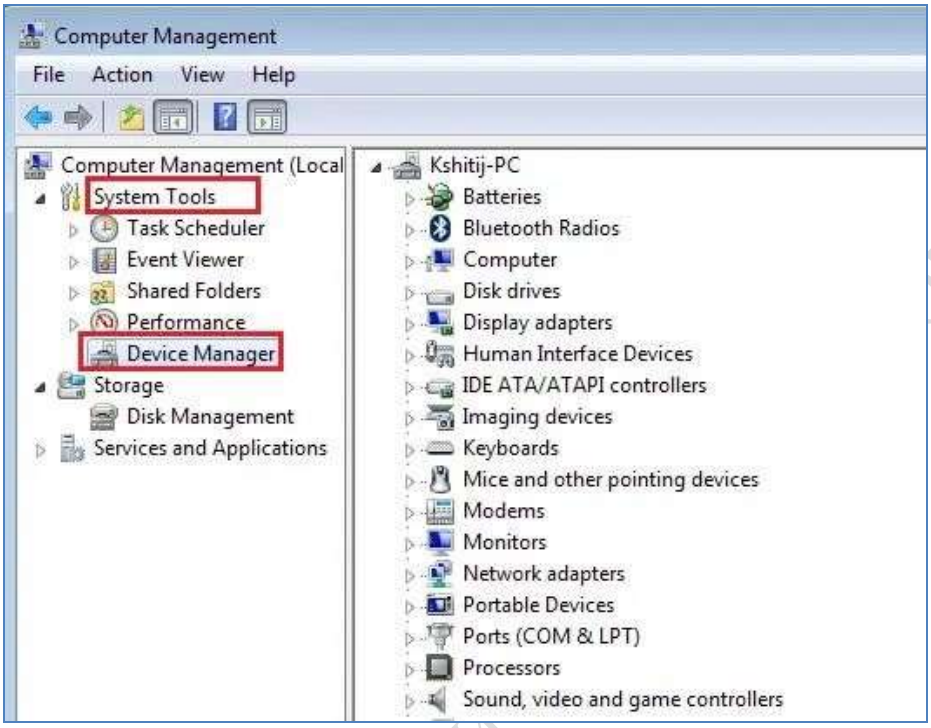


Часть 2. Настройка системы видеовывода

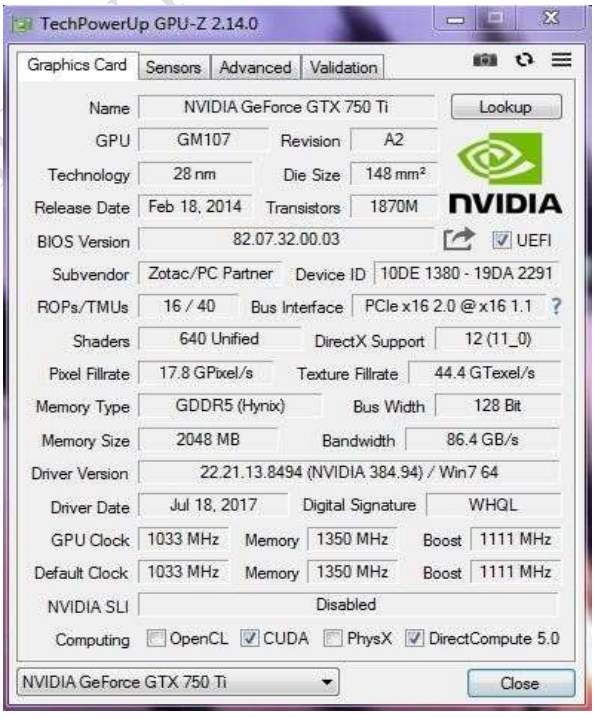
Видеосистема компьютера состоит из трех компонент: монитор (называемый также дисплеем); видеоадаптер; программное обеспечение (драйверы видеосистемы).

Видеоадаптер посылает в монитор сигналы управления яркостью лучей и синхросигналы строчной и кадровой развёрток. Монитор преобразует эти сигналы в

зрительные образы. А программные средства обрабатывают видеоизображения – выполняют кодирование и декодирование сигналов, координатные преобразования, сжатие изображений и др. Если необходимо получить дополнительную информацию, можно вызвать контекстное меню правой кнопкой мыши на объекте «Этот компьютер» или «Компьютер» и выберите команду « Управление», чтобы открыть служебную программу управления компьютером.



CPU-Z — бесплатная программа, которая помогает определить все комплектующие вычислительной системы. Скачать CPU-Z можно с официального сайта производителя. Можно загрузить специализированную утилиту TechPowerUp GPU-Z.



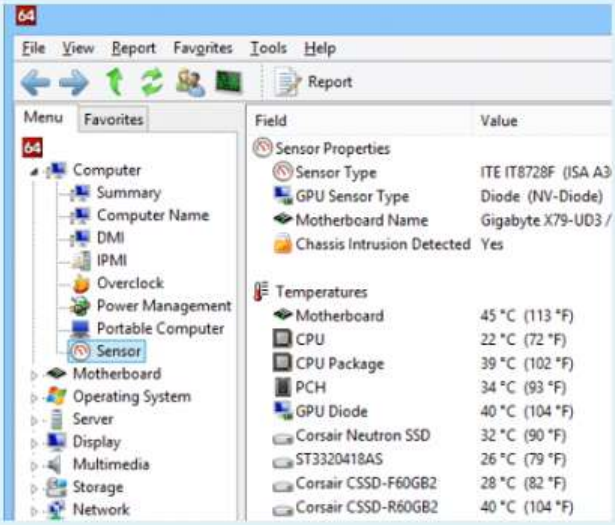
5 ТЕСТОВЫЕ ЗАДАНИЯ (примеры)

Укажите ошибку в наборе концептуальных требований для разработки планировщика процессов и задач операционной системы.

Выберите один ответ:

- ○ Максимизировать пропускную способность
- ○ Минимизировать расход энергии на организацию вычислительного процесса
- ○ Максимизировать среднее время отклика
- ○ Максимизировать использование ЦП
- ○ Минимизировать среднее время ожидания

Определите название утилиты обслуживания и/или диагностики устройства на изображении.



Выберите один ответ:

- ○ CPUz
- ○ AIDA64
- ○ MEMtest
- ○ CrystalDiskMark
- ○ Victoria

Какая организация ОС позволяет упростить работу пользователей и программистов в сетевых средах?

Выберите один ответ:

- ○ Распределенная организация.
- ○ Объектно-ориентированный подход.
- ○ Микроядерный подход.
- ○ Наличие нескольких прикладных сред.
- ○ Монолитное ядро.

Взгляните на изображение.

Программное обеспечение какого из сетевых узлов на нем изображено?

Выберите один ответ:

- ○ локальный интерфейс терминала POS
- ○ локальный интерфейс SmartTV
- ○ SSH-подключение к удаленной операционной системе
- ○ Локальная программа управления сетевым принтером
- ○ Веб-интерфейс сетевого маршрутизатора

Определите тип виртуализации на изображении.

Выберите один ответ:

- ○ Гипервизор II типа
- ○ Гипервизор I типа
- ○ Паравиртуализация
- ○ Изолированное рабочее пространство процессов (Portable version/Sandbox)
- ○ Виртуализация соотносительных операционных систем

Многие современные компании оставляют за собой право контроля кода управления устройством, переданного во владению пользователю. Какой из вендоров получил репутационные потери после примера, описанного далее:
Зачем взламывают пользовательские профили? Следствия в 99% случаев одинаковы: блокировка iPhone, iPad и компьютеров Mac "режимом пропажи" или их полное стирание в iCloud.

Выберите один ответ:

- ○ MicroSoft
- ○ Cisco
- ○ Apple
- ○ Samsung
- ○ Tesla

Какой компанией разработан Project xCloud?

Выберите один ответ:

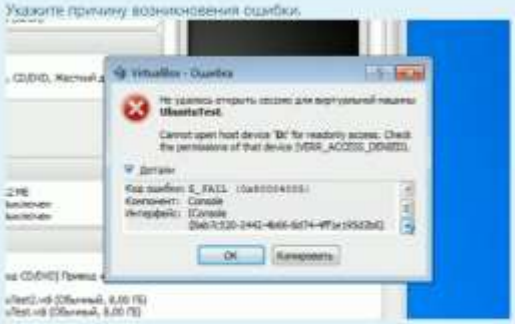
- MicroSoft
- Apple
- Huawei
- Samsung
- Google

Для чего предназначена команда **C:\>cls**?

Выберите один ответ:

- Переименование файла
- Очистить консоль (удалить историю выполнения команд из окна терминала)
- Вывести список запущенных процессов
- Показать дерево папок внутри папки html
- Справка по команде tree

Укажите причину возникновения ошибки.



Выберите один ответ:

- нарушение внутренней структуры файла или использование более новой версии гипервизора источника виртуальной машины
- для файла, содержащего образ диска виртуальной машины, установлены права только чтения
- отсутствие или повреждение исходного файла сохраненной сессии
- конфликт с действующим компонентом виртуализации Windows
- отключена или отсутствует аппаратная поддержка виртуализации

Какая из предложенных команд выведет на экран последнее по алфавиту имя файла в текущем каталоге?

Выберите один ответ:

- /home/larry/papers# ls | head -1 notes
- /home/larry/papers# ls | sort -r | head -1 notes
- /home/larry/papers# ls /usr/bin | more
- /home/larry/papers# ls | sort -r
- /home/larry/papers# ls /usr/bin | more | head -1 notes

Учреждение образования
«Гомельский государственный университет имени Франциска Скорины»

УТВЕРЖДАЮ
Проректор по учебной работе
ГГУ имени Ф. Скорины

_____ И.В. Семченко
_____
(дата утверждения)
Регистрационный № УД-_____/ уч.

Модуль «Основы информационных технологий»:
ОПЕРАЦИОННЫЕ СИСТЕМЫ

Учебная программа учреждения высшего образования по учебной дисциплине для специальности

1-53 01 02 Автоматизированные системы обработки информации

2021 г.

Учебная программа составлена на основе образовательного стандарта ОСВО 1-53 01 02-2021 г. и учебного плана ГГУ имени Ф.Скорины регистрационный № I 53-1-21/УП, дата утверждения 31.05.2021.

СОСТАВИТЕЛЬ:
А.В.Воруев, доцент кафедры АСОИ

Рецензенты:
1. Л.П. Авдашкова, доцент кафедры информационно-вычислительных систем УО «Белорусский торгово-экономический университет потребительской кооперации», к.ф.-м.н., доцент
2. А.Л. Самофалов, доцент кафедры общей физики УО «Гомельский государственный университет имени Франциска Скорины», к.ф.-м.н., доцент

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:

Кафедрой автоматизированных систем обработки информации
(протокол № 11 от 18.06.2021);

Научно-методическим советом Учреждения образования «ГГУ имени Ф.Скорины»
(протокол № 9 от 28.07.2021).

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Изучение дисциплины государственного компонента «Операционные системы» модуля «Основы информационных технологий» предусмотрено учебным планом подготовки специалистов специальности 1-53 01 02 – «Автоматизированные системы обработки информации».

Актуальность изучения дисциплины обусловлена высокой скоростью обновления технологической линейки электронных вычислительных систем и машин, а также смежных систем организации вычислительного процесса.

Целью курса «Операционные системы» является овладение студентами теоретическими и практическими основами использования вычислительной техники.

Операционная система является посредником (связующим звеном, интерфейсом) между ЭВМ, с одной стороны, и человеком (пользователем, программистом, оператором, инженером и т.д.) — с другой.

Знание механизмов организации вычислительного процесса на компьютере полезно для любого специалиста в области информационных технологий, а для программиста является обязательным.

Задачами курса являются:

- ознакомление с историей развития операционных систем, принципами работы и системным программирование;
- усвоение профессиональных навыков установки и настройки операционных систем;
- овладение основными принципами установки системного и прикладного программного обеспечения;
- анализ аппаратного обеспечения вычислительной системы посредством специализированного программного обеспечения;
- изучение методов программирования в современных операционных системах;
- формирование умений и навыков практического применения полученных знаний.

В результате изучения дисциплины:

Студент должен иметь представление:

- о типах современных вычислительных систем;
- об их составе и принципах работы составных частей;

Студент должен знать:

- виды технологий и режимы использования вычислительных систем;
- аппаратно-программные платформы современных вычислительных систем;
- мультипроцессорные конфигурации и элементы микропроцессорных систем;

Студент должен владеть:
- методами и средствами машинного обучения;

Студент должен уметь использовать:
- персональные вычислительные системы;
- универсальные и специализированные устройства ввода данных;
- твердотельные накопители;
- устройства печати.

После изучения дисциплины студент должен обладать следующими видами компетенций:

БПК-13 Управлять операционными системами, использовать методы планирования задач, синхронизации, администрирования и защиты информации.

Дисциплина государственного компонента «Операционные системы» изучается студентами 1 курса дневной, заочной и дистанционной форм обучения специальности 1-53 01 02 - «Автоматизированные системы обработки информации».

Дневная форма обучения: всего часов по плану – 104 (3 зач. ед.); аудиторное количество часов – 52, из них: лекции – 36, лабораторные занятия – 16. Курсовой проект.

Форма отчётности – зачет во 2 семестре.

Заочная, заочная сокращенная и дистанционная формы обучения: всего часов по плану – 104, аудиторное количество часов – 14, из них: лекции – 10, лабораторные занятия – 4. Курсовой проект.

Форма отчётности – контрольная работа и зачет во 2 семестре.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Раздел 1 Операционные системы в составе инструментов системного программного обеспечения

Тема 1.1 Назначение операционных систем

Эволюция операционных систем (ОС). Основные идеи построения ЭВМ разных поколений. Поколения ОС. Определение ОС, состав и функции операционной системы. Основные типы операционных систем. Однопрограммная (однозадачная) ОС. Системы последовательной пакетной обработки. Мультипрограммные системы пакетной обработки. Системы разделения времени. Системы реального времени. Системы виртуальных машин. Мультипроцессорная обработка. Сетевые операционные системы.

Тема 1.2 Интерфейсы операционных систем

Пользовательским интерфейсом называется набор приемов взаимодействия пользователя с приложением. Пользовательский интерфейс включает общение пользователя с приложением и язык общения. Пакетный интерфейс. Консольный ввод. Командный интерфейс. Простой графический интерфейс. Псевдографические системы. WIMP-интерфейс. WEB-интерфейс. Речевой интерфейс. SILK-системы. Интерфейс распознавания образов. Тактильный интерфейс. Семантический интерфейс.

Раздел 2 Формальный подход к описанию операционных систем

Тема 2.1 Функции операционных систем

Функции ОС по управлению ресурсами. Управление процессами. Управление памятью. Управление файлами и внешними устройствами. Защита данных и поддержка администрирования. Поддержка интерфейса прикладного программирования. Поддержка пользовательского интерфейса.

Архитектурные особенности операционных систем. Обобщенная структура ОС. Архитектура ОС на основе монолитного ядра. Особенности работы ядра в привилегированном режиме. Многослойная структура ОС. Машинно-зависимые компоненты ОС. Микроядерная архитектура.

Тема 2.2 Процессы и потоки

Понятия «процесс» и «поток». Создание процессов и потоков. Контекст процесса (потока). Дескриптор процесса (потока), состояния потока, операции над потоками (процессами). Завершение процесса и потока.

Тема 2.3 Алгоритмы планирования

Планирование и диспетчеризация потоков, вытесняющие и не вытесняющие алгоритмы планирования. Алгоритмы планирования, основанные на квантовании. Циклическое планирование. Многоуровневые очереди с обратными связями. Квантование с предпочтением потоков, с интенсивным вводом-выводом. Алгоритмы планирования на основе

приоритетов Динамические, статические, относительные и абсолютные приоритеты. Смешанные алгоритмы планирования. Планирование в системах реального времени. Моменты перепланировки. Назначение и типы прерываний. Внешние, внутренние и программные прерывания. Состояния процессора. Механизмы обработки прерываний. Приоритеты прерываний. Маскирование прерываний. Диспетчеризация и приоритеты прерываний в ОС. Системные вызовы. Схемы обработки и режимы выполнения системных вызовов.

Тема 2.4 Синхронизация процессов и потоков

Необходимость синхронизации и гонки. Понятие критической секции. Задачи синхронизации (взаимного исключения, «производитель-потребитель», «читатели-писатели», «клиент-официант», «обедающие философы»). Средства синхронизации потоков одного процесса: системы прерываний, блокирующие переменные и семафоры. Системные объекты ОС для синхронизации потоков разных процессов (поток, процесс, файл, семафоры, мьютексы, события, мониторы, сигналы, сообщения).

Тема 2.5 Тупиковые ситуации

Тупиковые ситуации и подходы к их разрешению: понятие тупика, условия возникновения тупиков, подходы к разрешению проблемы тупиков (предотвращение, обход, распознавание); дисциплины предотвращения тупиков; алгоритм банкира; модели и алгоритмы обнаружения тупиков.

Раздел 3 Ресурсы в составе операционных систем

Тема 3.1 Драйверы устройств

Понятие драйвера устройства. Установка драйверов в различных операционных системах. Порядок использования драйверов в системе. Электронная подпись драйверов. Совместимость драйверов.

Организация устройств ввода – вывода. Автоматизация процессов обмена информацией. Блочные и символьные устройства. Управление процессами ввода – вывода.

Тема 3.2 Настройка параметров работы устройств для взаимодействия с операционной системой

Способы управления и настройки устройств вычислительной системы. Организация взаимодействия между устройствами. Назначение и структура BIOS. Ресурсы BIOS. Функциональная характеристика устройства. Процедуры инициализации и POST-диагностики. Цифровая и звуковая индикация ошибок. Передача управления загрузчику операционной системы. Перепрошивка BIOS. Моддинг BIOS. Программа CMOS. Основная страница конфигурации. Расширенные страницы конфигурации. Оверклокинг. Защита CMOS на уровне пароля. Защита настроек. Ограничение доступа. Шифрование оборудования. Настройка устройств средствами современных операционных систем.

Тема 3.3 Управление памятью

Функции ОС по управлению памятью. Методы распределения памяти. Распределение памяти фиксированными и динамическими разделами. Перемещаемые разделы. Методы распределения памяти с использованием дискового пространства. Понятие виртуальной памяти. Виртуальное адресное пространство. Страничное распределение. Сегментное распределение. Сегментно-страничное распределение. Разделяемые сегменты памяти. Кэширование. Получение системной информации о виртуальной памяти. Состояние адресного пространства. Резервирование и освобождение регионов. Передача региону памяти и ее возврат. Атрибуты защиты памяти. Изменение атрибутов защиты памяти. Блокировка страниц в оперативной памяти. Сброс содержимого физической памяти

Иерархия запоминающих устройств. Кэш-память, принцип действия кэш-памяти. Кэширование данных. Размещение объектов в памяти. Куча и стек. Использование указателей. Динамическое выделение памяти.

Тема 3.4 Понятие виртуального ресурса

Понятие виртуального ресурса. Организация виртуального ресурса. Виртуальный процессор. Виртуальная память. Виртуальные каналы ввода – вывода. Виртуализация устройств в составе вычислительной системы. Виртуализация информационных потоков. Виртуализация реакции пользователей. Понятие «бот». Программные средства виртуализации.

Тема 3.5 Создание и управление виртуальными машинами

Виды гипервизоров. Гипервизоры типа 1 и типа 2. Разделяемы ресурсы виртуальных машин. Сетевое взаимодействие виртуальных машин. Гипервизор с открытым кодом KVM. Hyper-V. VMware. ESXi. Vagrant. Docker. Кластеры виртуализации. Балансировка нагрузки.

Раздел 4 Гетерогенное взаимодействие операционных систем

Тема 4.1 Операционные системы сетевых узлов

Понятие гетерогенности. Понятие мультиплатформенности. Модель сетевого взаимодействия ISO OSI. Протоколы сетевого взаимодействия. Пиринговое взаимодействие (p2p). Взаимодействие «клиент-сервер». Операционные системы клиентов. Веб-интерфейс и консольное управление. Операционные системы серверов. Операционные системы коммутирующих устройств. Операционные системы маршрутизаторов. Сбор статистик. Фильтрация сетевого трафика. Регистрация ошибок. Механизмы информирования.

Тема 4.2 Операционные системы IoT

Функционал операционных систем устройств IoT. Требования к операционным системам IoT: минимальное потребление памяти; энергоэффективность; возможности подключения к сети; аппаратно-независимые операции; требования к работе в реальном времени; требования

безопасности; экосистема разработки приложений. Концепция «туманных» вычислений. API операционных систем устройств IoT. Примеры операционных системы IoT: Ubuntu Core; RIOT; Contiki; TinyOS; Zephyr.

Тема 4.3 Процедуры обеспечения безопасности

Необходимые компоненты базовой локальной политики безопасности. Задачи по защите оборудования. Способы защиты данных. Методы обеспечения безопасности в беспроводных сетях. Распространенные профилактические меры по обеспечению безопасности. Обновление файлов подписей для программ защиты от вирусов и шпионского ПО. Установка пакетов обновлений и исправлений для операционных систем. Устранение проблем в обеспечении безопасности. Обзор процедуры устранения проблем. Определение распространенных проблем и решений.

Тема 4.4 Построение изолированных программных сред

Аппаратно изолированная среда. Использование защищенных сетевых структур в публичном облаке. Сетевые политики. DMZ. Доверительные отношения между сетевыми сегментами. Тоннели. Защищенные тоннели. Виртуализация узлов. Программно-изолированная среды при межпроцессном управлении операционной системы. Контейнерный подход. Понятие «песочницы»

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА (дневная форма обучения)

Номер раздела, темы, занятия	Название раздела, темы, занятия; перечень изучаемых вопросов	Количество аудиторных часов					Кол-во часов УСР	Формы контроля знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия	Иное		
1	2	3	4	5	6	7	8	9
1.	**ОПЕРАЦИОННЫЕ СИСТЕМЫ В СОСТАВЕ ИНСТРУМЕНТОВ СИСТЕМНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (4 Ч.)**	**4**						
1.1.	**Назначение операционных систем** 1. Эволюция операционных систем. 2. Классификация ОС. 3. Системы виртуальных машин.	**2**						
1.2	**Интерфейсы операционных систем** 1. Пакетный интерфейс. 2. Графический интерфейс. 3. Речевой интерфейс и интерфейс распознавания образов. 4. Тактильный интерфейс. 5. Семантический интерфейс.	**2**						
2.	**ФОРМАЛЬНЫЙ ПОДХОД К ОПИСАНИЮ ОПЕРАЦИОННЫХ СИСТЕМ (14 Ч.)**	**10**			**4**			
2.1	**Функции операционных систем** 1. Функции ОС по управлению ресурсами. 2. Управление памятью. 3. Управление файлами и внешними устройствами.	**2**						
2.2	**Процессы и потоки** 1. Понятия «процесс» и «поток». 2. Создание процессов и потоков. 3. Контекст и дескриптор процесса.	**2**						
2.3	**Алгоритмы планирования** 1. Циклическое планирование. 2. Многоуровневые очереди с обратными связями. 3. Квантование с предпочтением потоков.	**2**						Тест

1	2	3	4	5	6	7	8	9
2.4	**Синхронизация процессов и потоков** 1. Необходимость синхронизации и гонки. 2. Понятие критической секции. 3. Задачи синхронизации.	**2**			**4**			Защита отчета по лаб. работе
2.5	**Тупиковые ситуации** 1. Тупиковые ситуации. 2. Подходы к решению тупиковых ситуаций. 3. Алгоритмы обнаружения тупиков.	**2**						
3.	**РЕСУРСЫ В СОСТАВЕ ОПЕРАЦИОННЫХ СИСТЕМ (20 Ч.)**	**10**			**8**			
3.1	**Драйверы устройств** 1. Понятие драйвера устройства. 2. Установка драйверов в различных операционных системах. 3. Порядок использования драйверов в системе.	**2**						Реферат
3.2	**Настройка параметров работы устройств для взаимодействия с операционной системой** 1. Способы управления и настройки устройств в вычислительных системах. 2. Назначение и структура BIOS. 3. Программа CMOS. 4. Настройка устройств средствами современных операционных систем.	**2**			**4**			Защита отчета по лаб. работе
3.3	**Управление памятью** 1. Функции ОС по управлению памятью. 2. Методы распределения памяти. 3. Иерархия запоминающих устройств.	**2**						
3.4	**Понятие виртуального ресурса** 1. Организация виртуального ресурса. 2. Виртуальный процессор. 3. Виртуальная память.	**2**						
3.5	**Создание и управление виртуальными машинами** 1. Гипервизоры типа 1 и типа 2. 2. Разделяемы ресурсы виртуальных машин. 3. Сетевое взаимодействие виртуальных машин.	**2**			**4**			Защита отчета по лаб. работе
4.	**ГЕТЕРОГЕННОЕ ВЗАИМОДЕЙСТВИЕ ОПЕРАЦИОННЫХ СИСТЕМ (16 Ч.)**	**8**			**4**			
4.1	**Операционные системы сетевых узлов** 1. Веб-интерфейс и консольное управление. 2. Фильтрация сетевого трафика. 3. Протоколы сетевого взаимодействия.	**2**						

1	2	3	4	5	6	7	8	9
4.2	**Операционные системы IoT** 1. Функционал операционных систем устройств IoT. 2. Концепция «туманных» вычислений. 3. Энергообеспечение устройств IoT. 4. API операционных систем устройств IoT.	**2**						Реферат
4.3	**Процедуры обеспечения безопасности** 1. Задачи по защите оборудования на уровне операционных систем. 2. Способы защиты данных. 3. Профилактические меры по обеспечению безопасности.	**2**			**4**			Защита отчета по лаб. работе
4.4	**Построение изолированных программных сред** 1. Аппаратно изолированная среда. 2. Использование защищенных сетевых структур в публичном облаке. 3. Контейнерный подход.	**2**						
	Всего по дисциплине	**32**			**16**			зачет

Доцент кафедры АСОИ А.В.Воруев

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА (заочная форма обучения, заочная интегрированная форма обучения на основе среднего специального образования, дистанционная форма обучения)

Номер раздела, темы, занятия	Название раздела, темы, занятия; перечень изучаемых вопросов	Количество аудиторных часов					Кол-во часов УСР	Формы контроля знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия	Иное		
1	2	3	4	5	6	7	8	9
1.	**ОПЕРАЦИОННЫЕ СИСТЕМЫ В СОСТАВЕ ИНСТРУМЕНТОВ СИСТЕМНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (2 Ч.)**	**2**						
1.1.	**Назначение операционных систем** 1. Эволюция операционных систем. 2. Классификация ОС. 3. Системы виртуальных машин.	**2**						Тест
1.2	**Интерфейсы операционных систем** 1. Пакетный интерфейс. 2. Графический интерфейс. 3. Речевой интерфейс и интерфейс распознавания образов. 4. Тактильный интерфейс. 5. Семантический интерфейс.	Самостоятельное изучение						
2.	**ФОРМАЛЬНЫЙ ПОДХОД К ОПИСАНИЮ ОПЕРАЦИОННЫХ СИСТЕМ (2 Ч.)**	**2**						
2.1	**Функции операционных систем** 1. Функции ОС по управлению ресурсами. 2. Управление памятью. 3. Управление файлами и внешними устройствами.	Самостоятельное изучение						
2.2	**Процессы и потоки** 1. Понятия «процесс» и «поток». 2. Создание процессов и потоков. 3. Контекст и дескриптор процесса.	Самостоятельное изучение						

1	2	3	4	5	6	7	8	9
2.3	**Алгоритмы планирования** 1. Циклическое планирование. 2. Многоуровневые очереди с обратными связями. 3. Квантование с предпочтением потоков.	**2**						Тест
2.4	**Синхронизация процессов и потоков** 1. Необходимость синхронизации и гонки. 2. Понятие критической секции. 3. Задачи синхронизации.	Самостоятельное изучение						
2.5	**Тупиковые ситуации** 1. Тупиковые ситуации. 2. Подходы к решению тупиковых ситуаций. 3. Алгоритмы обнаружения тупиков.	Самостоятельное изучение						
3.	**РЕСУРСЫ В СОСТАВЕ ОПЕРАЦИОННЫХ СИСТЕМ (8 Ч.)**	**4**			**4**			
3.1	**Драйверы устройств** 1. Понятие драйвера устройства. 2. Установка драйверов в различных операционных системах. 3. Порядок использования драйверов в системе.	Самостоятельное изучение						
3.2	**Настройка параметров работы устройств для взаимодействия с операционной системой** 1. Способы управления и настройки устройств в вычислительных системах. 2. Назначение и структура BIOS. 3. Программа CMOS. 4. Настройка устройств средствами современных операционных систем.	Самостоятельное изучение						
3.3	**Управление памятью** 1. Функции ОС по управлению памятью. 2. Методы распределения памяти. 3. Иерархия запоминающих устройств.	Самостоятельное изучение						
3.4	**Понятие виртуального ресурса** 1. Организация виртуального ресурса. 2. Виртуальный процессор. 3. Виртуальная память.	**2**						
3.5	**Создание и управление виртуальными машинами** 1. Гипервизоры типа 1 и типа 2. 2. Разделяемы ресурсы виртуальных машин. 3. Сетевое взаимодействие виртуальных машин.	**2**			**4**			Защита отчета по лаб. работе
4.	**ГЕТЕРОГЕННОЕ ВЗАИМОДЕЙСТВИЕ ОПЕРАЦИОННЫХ СИСТЕМ (16 Ч.)**	**2**						
4.1	**Операционные системы сетевых узлов** 1. Веб-интерфейс и консольное управление. 2. Фильтрация сетевого трафика.	**2**						Тест

1	2	3	4	5	6	7	8	9
	3. Протоколы сетевого взаимодействия.							
4.2	**Операционные системы IoT** 1. Функционал операционных систем устройств IoT. 2. Концепция «туманных» вычислений. 3. Энергообеспечение устройств IoT. 4. API операционных систем устройств IoT.	Самостоятельное изучение						
4.3	**Процедуры обеспечения безопасности** 1. Задачи по защите оборудования на уровне операционных систем. 2. Способы защиты данных. 3. Профилактические меры по обеспечению безопасности.	Самостоятельное изучение						
4.4	**Построение изолированных программных сред** 1. Аппаратно изолированная среда. 2. Использование защищенных сетевых структур в публичном облаке. 3. Контейнерный подход.	Самостоятельное изучение						
	Всего по дисциплине	**10**			**4**			зачет

Доцент кафедры АСОИ А.В.Воруев

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

ПРИМЕРНЫЙ ПЕРЕЧЕНЬ ТЕМ ЛАБОРАТОРНЫХ ЗАНЯТИЙ

1. Синхронизация процессов и потоков.
2. Настройка параметров работы устройств для взаимодействия с операционной системой.
3. Создание и управление виртуальными машинами.
4. Процедуры обеспечения безопасности.

ПРИМЕРНЫЙ ПЕРЕЧЕНЬ НЕОБХОДИМОГО ОБОРУДОВАНИЯ И КОМПЬЮТЕРНЫХ ПРОГРАММ

1 Класс современных персональных компьютеров.

2 Материалы электронного курса «Основы информационных технологий» международного образовательного проекта Cisco Networking Academy .

3 Программная система моделирования и виртуализации Oracle Virtual BOX.

КУРСОВОЙ ПРОЕКТ

Разработать программный комплекс сбора информации из вычислительной системы и сохранением ее в общий файл (или базу данных) на сервере.

Оформляется курсовой проект согласно положения СТП 04-2011, можно использовать шаблон.

Структура курсового проекта:

- Титульный лист.
- Реферат.
- Содержание.
- Введение.
- 1 Описание теоретических сведений.
- 2 Выбор программных средств реализации проекта.
- 3 Реализация проекта.
- Заключение.
- Список использованных источников.
- Приложение А.

В таблицах указаны пункты общего задания, которые необходимо выполнить коллективу для операционных систем семейства Windows (x86 и x64) или Linux. Программные средства разрабатываются на любом подходящем языке программирования по выбору исполнителя.

Задание для коллективной разработки	
1.	Определение полной информации о ЦП средствами
2.	Определение полной информации о BIOS средствами
3.	Определение полной информации о разделах HDD средствами
4.	Определение полной информации об HDD средствами
5.	Определение полной информации об использовании клавиатуры средствами
6.	Определение полной информации о системной плате средствами
7.	Определение полной информации об использовании mouse средствами
8.	Определение полной информации о видеокарте средствами
9.	Определение полной информации о мониторе средствами
10.	Определение полной информации о сетевых адаптерах средствами
11.	Определение информации о запущенных приложениях средствами
12.	Сбор всех данных в один текстовый файл и транспортировка его на сервер в указанную папку
13.	Сбор информации о дате и времени запуска и завершения Windows (подсчет времени работы ПК)
14.	Сбор информации о дате и времени входа и выхода из системы определенного пользователя (подсчет времени сколько каждый пользователь провел в системе)

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

ОСНОВНАЯ

1. Власов, Ю. В. Администрирование сетей на платформе MS Windows Server : учебное пособие / Ю. В. Власов, Т. И. Рицкова. – Москва: Интернет-ун-т Информационных Технологий : БИНОМ. Лаборатория знаний, 2014 . – 384 с.
2. Гордеев, А. В. Операционные системы : учебник для вузов / А. В. Гордеев. – СПб. : Питер, 2005. – 416 с.
3. Иртегов, Д. В. Введение в операционные системы / Д. В. Иртегов. – СПб. : БХВ – Петербург, 2002. – 624 с.
4. Молчанов, А. Ю. Системное программное обеспечение : учебник для вузов / А. Ю. Молчанов. – СПб. : Питер, 2010. – 410 с.
5. Назаров, С. В. Современные операционные системы : учебное пособие / С. В. Назаров . – Москва : Национальный Открытый Ун-т Институт : БИНОМ , 2013. – 367 с.
6. Олифер, В. Г. Сетевые операционные системы: Учебник для вузов. 2-е изд. / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2009. – 669 с.: ил.
7. Станек, У. Р. Windows 7. Справочник администратора / Уильям Р. Станек. – М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2010 – 720 с.
8. Станек, У. Р. Windows PowerShell. Справочник администратора / Уильям Р. Станек. – М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2010 – 416 с.
9. Столлингс, В. Операционные системы / В. Столингс ; пер. с англ. – М. : Издат. дом «Вильямс», 2004. – 848 с.
10. Таненбаум, Э. Современные операционные системы / Э. Таненбаум. – СПб.: Питер, 2013. – 1120 с.

ДОПОЛНИТЕЛЬНАЯ

11. Гордеев, А. В. Системное программное обеспечение / А. В. Гордеев, А. Ю. Молчанов. – СПб. : Питер, 2002. – 736 с.
12. Дейтел, П. Как программировать на С / П. Дейтел, Х. Дейтел. – М. : Бином, 2005. – 912 с.
13. Дейтел, Х. Операционные системы. Распределение системы, сети, безопасность/ Х. Дейтел, П. Дейтел, Д. Чофиес. – М. : Бином, 2009. – 704 с.
14. Дейтел, Х. Операционные системы. Часть 1. Основы и принципы/ Х. Дейтел, П. Дейтел, Д. Чофиес. – М. : Бином, 2013. – 1024 с.
15. Дейтел, Х. Операционные системы. Часть 2. Распределенные системы, сети, безопасность/ Х. Дейтел, П. Дейтел, Д. Чофиес. – М. : Бином, 2013. – 704 с.
16. Джонс, В. Освой самостоятельно С за 21 день / В. Джонс, П. Эйткен. – М. : Издат. дом «Вильямс», 2003. – 800 с.
17. Кип, Р. И. Язык Ассемблера для процессоров Intel / Р. И. Кип. – М. : Издат. дом «Вильямс», 2005. – 912 с.

18. Колобко, И. Администрирование сетей Windows с помощью сценариев. – СПб. : BNV, 2007. – 306 с.

19. Колобко, И. Справочник системного администратора по программированию Windows. – СПб. : BNV, 2009. – 576 с.

20. Компиляторы: принципы, технологии и инструменты / А. Ахо [и др.]. – М. : Издат. дом «Вильямс», 2008. – 768 с.

21. Корнелл, Г. Java 2. Т. 1: Основы. Библиотека профессионала / Г. Корнелл, К. Хорстманн ; пер с англ. – М. : Издат. дом «Вильямс», 2008. – 816 с.

22. Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл. – СПб. : Питер, 2008. – 896 с.

23. Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл. – СПб. : Питер, 2012. – 896 с.

24. Моли, Б. Unix/Linux : теория и практика программирования / Б. Моли. – М. : КУДИЦ-ОБРАЗ, 2004. – 576 с.

25. Операционная система реального времени QNX Neutrino 6.3. Руководство пользователя – СПб. : БХВ – Петербург, 2009. – 480 с.

26. Операционная система реального времени QNX Neutrino 6.3. Системная архитектура – СПб. : БХВ – Петербург, 2005. – 336 с.

27. Пирогов, В. Ассемблер для Windows / В. Пирогов. – Киев : BHV, 2007. – 896 с.

28. Прайс, Дж. Visual C# . NET / Дж. Прайс, М. Гандэрлой. – Киев : ВЕК ; М. : Энтроп, 2004. – 960 с.

29. Прата, С. Язык программирования С. Лекции и упражнения / С. Прата. – М. : Издат. дом «Вильямс», 2006. – 960 с.

30. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C# / Дж. Рихтер ; пер с англ. – СПб.: Питер, 2008. – 856 с.

31. Рихтер, Дж. Windows для профессионалов : создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows NT / Дж. Рихтер ; пер. с англ. – СПб. : Питер; М. : Издат.-торг. дом «Русская редакция», 2001.– 752 с.

32. Рихтер, Дж. Программирование серверных приложений для Microsoft Windows 2000 / Дж. Рихтер, Кларк Дж. Д.; пер. с англ. – СПб. : Питер; М. : Издат.-торг. дом «Русская редакция», 2001. – 592 с.

33. Роббинс, А. Linux : программирование в примерах / А. Роббинс. – М. : КУДИЦ-ОБРАЗ, 2005. – 656 с.

34. Руссинович, М. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000 / М. Руссинович, Д. Соломон ; пер. с англ. – 4-е изд. – М. : Издат.-торг. дом «Русская Редакция» ; СПб. : Питер, 2005. – 992 с.

35. Седжвик, Р. Фундаментальные алгоритмы на C++. Ч. 1–4 / Р. Седжвик ; пер. с англ. – Киев : ДиаСофт, 2002. – 687 с.

36. Станек, У. Р. Командная строка Microsoft Windows : справочник администратора / У. Р. Станек ; пер. с англ. – М. : Издат.-торг. дом «Русская редакция», 2004. – 480 с.

37. Стивенс, У. UNIX : взаимодействие процессов / У. Стивенс. – СПб. : Питер, 2002. – 576 с.

38. Стивенс, У. UNIX : разработка сетевых приложений / У. Стивенс. – СПб. : Питер, 2003. – 1088 с.

39. Троелсен, Э. Язык программирования C# 2005 и платформа .NET / Э. Троелсен. – М. : Издат. дом «Вильямс», 2007. – 1168 с.

40. Харт, Д. Системное программирование в среде Windows / Д. Харт. – М. : Издат. дом «Вильямс», 2005. – 592 с.

41. Хэзфилд, Р. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений / Р. Хэзфилд, Л. Кирби. – Киев : ДиаСофт, 2001. – 736 с.

42. Шилдт, Г. Полный справочник по С / Г. Шилдт. – М. : Издат. дом. «Вильямс», 2007. – 704 с.

ЭЛЕКТРОННЫЕ РЕСУРСЫ

1 Свободная энциклопедия ВикипедиЯ [Электронный ресурс]. – 2021. – Режим доступа: http://ru.wikipedia.org. – Дата доступа: 12.03.2021.

2 Интернет университет информационных технологий [Электронный ресурс]. – 2021. – Режим доступа: http://www.intuit.ru. – Дата доступа: 12.03.2021.

3 Информационно-справочный портал технической информации Хабрахабр [Электронный ресурс]. – 2021. – Режим доступа: http://citforum.ru. – Дата доступа: 12.03.2021.

4 Виртуальный музей истории вычислительной техники [Электронный ресурс]. – 2021. – Режим доступа: http://computerhistory.narod.ru/. – Дата доступа: 12.03.2021.