МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ

Приложение 2
к рабочей программе дисциплины

# ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

## Операционные системы

*(наименование дисциплины(модуля)*

Направление подготовки / специальность

**09.03.03 Прикладная информатика**
*(код и наименование)*

Направленность (профиль)/специализация

**Прикладная информатика на железнодорожном транспорте**
*(наименование)*

Содержание

1. Пояснительная записка.
2. Типовые контрольные задания или иные материалы для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих уровень сформированности компетенций.
3. Методические материалы, определяющие процедуру и критерии оценивания сформированности компетенций при проведении промежуточной аттестации.

## 1. Пояснительная записка

Цель промежуточной аттестации – оценивание промежуточных и окончательных результатов обучения по дисциплине, обеспечивающих достижение планируемых результатов освоения образовательной программы.

Перечень компетенций, формируемых в процессе освоения дисциплины

| Код и наименование компетенции |
|---|
| ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем; |
| ОПК-5.1 Администрирует аппаратное обеспечение информационных и автоматизированных систем |
| ОПК-5.2 Инсталлирует программное и аппаратное обеспечение информационных и автоматизированных систем |
| ОПК-9 Способен осваивать методики использования программных средств для решения практических задач |
| ОПК-9.2 Разрабатывает методики использования программных средств |

Результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

| Код и наименование индикатора достижения компетенции | Результаты обучения по дисциплине | Оценочные материалы |
|---|---|---|
| ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем | Обучающийся знает: архитектуру, состав и стандарты взаимодействия аппаратных модулей современных информационных и автоматизированных систем; | |
| | Обучающийся умеет: выполнять комплекную настройку аппаратного обеспечения современных информационных и автоматизированных систем; | |
| | Обучающийся владеет:навыками администрирования аппаратного обеспечения информационных и автоматизированных систем; | |
| ОПК-5.1 Администрирует аппаратное обеспечение информационных и автоматизированных систем | Обучающийся знает: архитектуру, состав и стандарты взаимодействия модулей современных операционных систем; | |
| | Обучающийся умеет: выполнять администрирование операционных систем и системного программного обеспечения; | |
| | Обучающийся владеет: навыками инсталляции программного и аппаратного обеспечения информационных и автоматизированных систем; | |
| ОПК-5.2 Инсталлирует программное и аппаратное обеспечение информационных и | Обучающийся знает: методики использования программных средств для решения практических задач | |

| | | |
|---|---|---|
| автоматизированных систем | | |
| | Обучающийся умеет: разрабатывать методики использования программных средситв для решения практических задач. | |
| | Обучающийся владеет: технологиями эффективного использования программных средств для решения практических задач | |
| ОПК-9 Способен осваивать методики использования программных средств для решения практических задач | Обучающийся знает: структуру данных, системы хранения и управления данными, архитектуру и принципы организации систем. | |
| | Обучающийся умеет: разрабатывать информационно-логическую структуру приложения. | |
| | Обучающийся владеет: способностью применять современные методы проектирования программного обеспечения. | |
| ОПК-9.2 Разрабатывает методики использования программных средств | Обучающийся знает:методологию проектирования баз данных, теорию функциональных на отношениях | |
| | Обучающийся умеет: программировать приложения в архитектуре клиент-сервер, приводить отношения к нормальным формам. | |
| | Обучающийся владеет: способностью обосновывать выбор программного обеспечения и разрабатывать концептуальную и логическую модель данных. | |

Промежуточная аттестация (экзамен) проводится в одной из следующих форм:
1) ответ на билет, состоящий из теоретических вопросов и практических заданий;
2) выполнение заданий в ЭИОС СамГУПС.

## 2. Типовые[1] контрольные задания или иные материалы для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих уровень сформированности компетенций

### 2.1 Типовые вопросы (тестовые задания) для оценки знаниевого образовательного результата

Проверяемый образовательный результат

| Код и наименование индикатора достижения компетенции | Образовательный результат |
|---|---|
| ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем; | Обучающийся знает: основные компоненты электронно - образовательной среды СамГУПС, доступные для обучающихся, основные системы видеоконференцсвязи ЭИОС, возможности ЭИОС для синхронного и асинхронного взаимодействия в рамках образовательного процесса, доступные в ЭИОС электронные библиотеки. Основные сервисы MicrosoftOffice 365, интегрированные в ЭИОС университета. Основные онлайн-сервисы и площадки, используемые в процессе самообразования |
| ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем | Обучающийся умеет: получать доступ к учебным планам, рабочим программам дисциплин (модулей), практик, к изданиям электронных библиотечных систем и электронным образовательным ресурсам, указанным в рабочих программах, использовать возможности систем видеоконференцсвязи для учебной (научной) работе и самообразования, с использованием средств ЭИОС, участвовать в проведении всех видов занятий, процедур оценки результатов обучения, реализация которых предусмотрена с применением электронного обучения, дистанционных образовательных технологий. Формировать свое электронное портфолио, в том числе сохранять свои работы, рецензий и оценки на них. Устанавливать на мобильные устройства сервисы ЭИОС университета, приложения MicrosoftOffice 365 и использовать их в учебной (научной) работе и самообразовании |
| ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем | Обучающийся владеет: навыками синхронного и (или) асинхронного взаимодействия посредством сети "Интернет" с использованием средств ЭИОС между участниками образовательного процесса. Навыками фиксации хода образовательного процесса, результатов промежуточной аттестации и результатов освоения программы бакалавриата в своем портфолио. Навыками использования сервисов ЭИОС университета, приложениями MicrosoftOffice 365 в процессе учебной (научной) работы и самообразовании |

[1]Приводятся типовые вопросы и задания. Оценочные средства, предназначенные для проведения аттестационного мероприятия, хранятся на кафедре в достаточном для проведения оценочных процедур количестве вариантов. Оценочные средства подлежат актуализации с учетом развития науки, образования, культуры, экономики, техники, технологий и социальной сферы. Ответственность за нераспространение содержания оценочных средств среди обучающихся университета несут заведующий кафедрой и преподаватель – разработчик оценочных средств.

## 2.2. Примерные задания

### Практическое занятие № 1.
### Выполнение действий с компонентами интерфейса пользователя.

**Учебная цель:**
Закрепление полученных теоретических знаний по теме «Интерфейс пользователя».

**Учебные задачи:**
1. Изучить различные виды интерфейсов пользователей
2. Научиться работать в популярных видах интерфейсов.

### Задачи практического занятия №1

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу с 2 видами интерфейса.
3. Подготовить отчет по практической работе

### Обеспеченность занятия (средства обучения)

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

### Краткие теоретические и учебно-методические материалы по теме практического занятия

Как любое техническое устройство, компьютер обменивается информацией с человеком посредством набора определенных правил, обязательных как для машины, так и для человека. Эти правила в компьютерной литературе называются интерфейсом. Интерфейс может быть понятным и непонятным, дружественным и нет. К нему подходят многие прилагательные. Но в одном он постоянен: он есть, и никуда от него не денешься.

**Интерфейс** - это правила взаимодействия операционной системы с пользователями, а также соседних уровней в сети ЭВМ. От интерфейса зависит технология общения человека с компьютером.

**Командный интерфейс** называется так по тому, что в этом виде интерфейса человек подает "команды" компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки.

При этой технологии в качестве единственного способа ввода информации от человека к компьютеру служит клавиатура, а компьютер выводит информацию человеку с помощью алфавитно-цифрового дисплея (монитора). Эту комбинацию (монитор + клавиатура) стали называть терминалом, или консолью.

**WIMP - интерфейс** (Window - окно, Image - образ, Menu - меню, Pointer - указатель). Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов - меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается "опосредственно", через графические образы.

**SILK - интерфейс** (Speech - речь, Image - образ, Language - язык, Knowlege - знание). Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный "разговор" человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы.

Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяют в основном для военных целей.

**Биометрическая технология** ("Мимический интерфейс".) возникла в конце 90-х годов XX века и на момент написания книги еще разрабатывается. Для управления компьютером используется выражение лица человека, направление его взгляда, размер зрачка и другие признаки. Для идентификации пользователя используется рисунок радужной оболочки его глаз, отпечатки пальцев и другая уникальная информация. Изображения считываются с цифровой видеокамеры, а затем с помощью специальных программ распознавания образов из этого изображения выделяются команды. Эта технология, по-видимому, займет свое место в программных продуктах и приложениях, где важно точно идентифицировать пользователя компьютера.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое интерфейс?
2. Какие виды интерфейсов существуют?

**Задания для практического занятия №1**

1. Задание 1:
   a) Запустите Проводник, в списке дисков и папок выберите Диск E:\.
   b) Задайте вид отображения папок и файлов в окне Проводника в виде таблицы, для чего в меню Вид выберите опцию Таблица.
   c) Отсортируйте размещение значков папок и файлов по возрастанию даты последнего изменения, для чего щелкните в правой части окна Проводника на заголовке графы Изменен.
   d) Упорядочите размещение значков папок и файлов по алфавиту имен, для чего щелкните в правой части окна Проводника на заголовке графы Имя.
   e) Создайте в корневой директории диска E:\ папку с именем 1111.
   f) Создайте в папке 1111 текстовый файл Пример1.txt.
   g) Создайте папку 2222 в корневой директории диска E:\ и скопируйте в нее файл Пример1.txt из папки 1111.
   h) Переименуйте файл Пример1.txt в папке 2222 на диске E:\ в файл Пример2.txt.
   i) Закройте окно Проводника Windows.
2. Задание 1:
   a) Запустите Командную строку.
   b) Измените текущие время и дату на компьютере.
   c) Измените цвет командной строки.
   d) Создайте в корневой директории диска E:\ папку с именем Dir1.
   e) Создайте в папке Dir1 папки с именами Dir11 и Dir12.
   f) Создайте в папке Dir11 текстовый файл 1.txt.
   g) Скопируйте в папку Dir12 файл 1.txt из папки Dir11.
   h) Переименуйте файл 1.txt в папке Dir12 в файл 2.txt.

**Инструкция по выполнению заданий практического занятия №1**

1) Внимательно прочитать задание.

2) При работе в командной строке внимательно вводите команды и названия файлов и папок.

3) При написании отчета не забудьте вставить скриншоты вашей работы.

**Методика анализа результатов, полученных в ходе практического занятия**

После выполнения задания, следует еще раз все внимательно проверить на наличие ошибок, особенно это касается команд и написания названий файлов и папок.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.

2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

**Образец отчета по практическому занятию.**

**Отчет**
по практическому занятию № 1.
Работа и особенности логических элементов и схем ЭВМ.

Учебная цель:
Закрепление полученных теоретических знаний по теме «Интерфейс пользователя».

Учебные задачи:
1. Изучить различные виды интерфейсов пользователей
2. Научиться работать в популярных видах интерфейсов.

Задание 1.
Открыть Проводник Windows (рис.1).

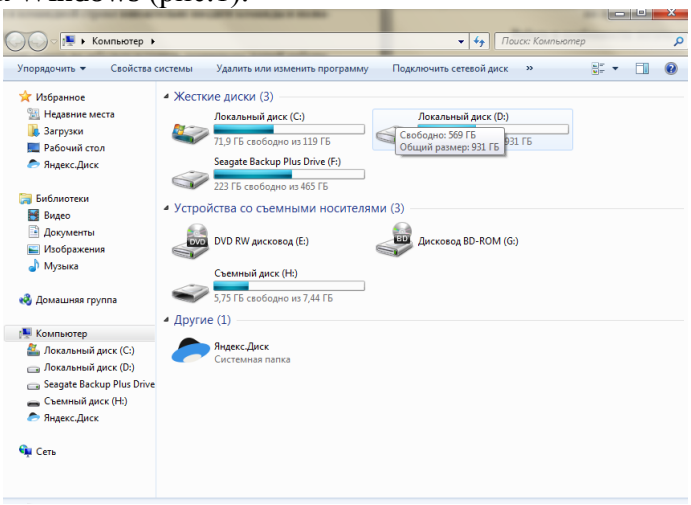


Рисунок 1. Проводник Windows

Задание 2.
Открыть командную строку (рис.2).

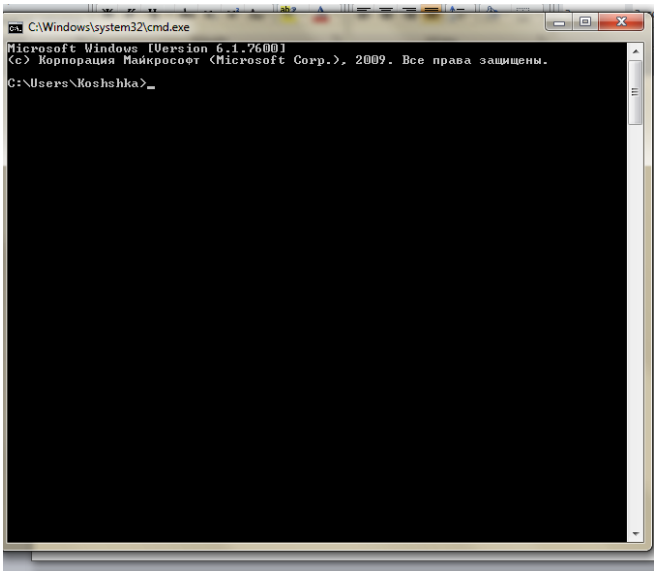


Рисунок 2. Командная строка Windows.

Вывод: В ходе проделанной работы я научился(-лась) ...

**Практическое занятие № 2.**
**Планирование процессов**

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Планирование и диспетчеризация процессов».

**Учебные задачи:**

1. Изучить планирование процессов

**Задачи практического занятия №2**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по практическому заданию.
4. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Процесс (process) это программа пользователя при ее выполнении. При своей работе операционная системы исполняет множество классов программ: пакетные задания; пользовательские программы в режиме разделения времени; системные программы и процессы.

Операционная система при управлении процессами обеспечивает их поочередное выполнение. Эту задачу решает планировщик ОС.

В операционной системе диспетчеризация процессов выполняется обычно несколькими планировщиками, каждый из которых имеет свою периодичность вызовов и свою определенную задачу, которую он решает.

Долговременный планировщик (планировщик заданий) определяет, какие процессы должны быть перемещены в очередь готовых процессов.

Кратковременный планировщик (планировщик процессора) – определяет, какие процессы должны быть выполнены следующими и каким процессам должен быть предоставлен процессор.

Для реализации режима разделения времени в систему может быть добавлен также планировщик откачки и подкачки процессов, определяющий, какие пользовательские процессы должны быть подкачаны в память или откачаны на диск.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое процесс?
2. Что такое планировщик процесса?
3. Какими бывают планировщики процессов?

**Задания для практического занятия №2**

1. Нарисовать модель процесса с пятью состояниями.

**Инструкция по выполнению заданий практического занятия №2**

1. Внимательно прочитать задание.
2. Приступить к выполнению задания, после тщательного анализа.

**Методика анализа результатов, полученных в ходе практического занятия**

Внимательно проверить все состояния процесса, проследить путь выполнения процесса.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 3.
## Создание файловой структуры диска.

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Создание файловой структуры диска».

**Учебные задачи:**

1. Изучить виды и типы файловых структур диска.
2. Научиться создавать и работать в файловой структуре диска.

**Задачи практического занятия №3**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить задание по работе в файловой структуре.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программноеобеспечение: ОС Windows, MS Word, FDiskили EASEUS Partition Manager.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

**Файл** – поименованная область на диске или другом носителе информации. В файлах могут храниться тексты программ, документы, готовые к выполнению программы и любые другие данные .

**Каталог** – это специальное место на диске, в котором хранятся имена файлов, сведения о размерах файлов, времени их последнего обновления, атрибуты (свойства) файлов и т.д. Если в каталоге хранится имя файла, то говорят, что этот файл находится в данном каталоге.

В операционных системах Windows папки и файлы образуют на дисках иерархическую **файловую структуру**. Необходимо знать, что понятия папка и каталог – это одно и тоже. Организация файловой структуры очень проста. Файлы находятся в папках. Папки вложены в другие папки, более высокого уровня. Папка самого высокого уровня называется корневой – она одна на каждом диске. Назначение файловой структуры – обеспечить однозначное отыскание любого файла, если известно его имя и путь поиска. Путь поиска начинается с корневой папки (ее имя совпадает с обозначением диска) и далее ведет через все вложенные папки к той папке, где находится разыскиваемый файл. Создание и обслуживание файловой структуры – это одна из основных функций операционной системы. Подводя итог, можно сказать, что файловая структура – это расположение файлов на диске в каталогах.

**Каталоговая структура** – это способ расположение папок (каталогов) на жестком диске.

При создании файлов и каталогов необходимо учитывать следующее правило - в одном каталоге нельзя создать два файла с одинаковыми именами, а в разных каталогах –можно.

**Логический диск** – это выделенная часть жесткого диска, имеющая буквенное обозначение. Жесткий диск может быть разбит на несколько логических дисков (например,

жесткий диск разбит на два логических диска – то мы видим при работе с компьютером диски A, C, D). Если логических дисков нет, то на компьютере обычно есть диски A и C. Основная причина разбития жесткого диска на несколько логических – удобство хранения и работы с файлами и папками.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое Файл?
2. Что такое Каталог?
3. Что такое Файловая структура?
4. Что из себя представляет логический диск?

**Задания для практического занятия №3**

1. Узнать тип файловой системы на вашем ПК.
2. Разбить жесткий диск системы на 2 логических диска.
3. Отформатировать новый логический диск, создав на нем файловую систему, отличную от прежней.
4. Сравнить плюсы и минусы файловых систем, использующихся на логических дисках.

**Инструкция по выполнению заданий практического занятия №3**

1. Внимательно прочитать задание.
2. Выполнить задания строго в соответствии с их порядком.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

При выполнении работы обратите внимание на детальное описание плюсов и минусов файловых систем, обязательно сделайте вывод о целесообразности использования той или иной файловой системы, а также выберете, которая, на ваш взгляд, наиболее удобна в работе.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 4
## Управление дисками и файловыми системами в Windows

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Создание файловой структуры диска».

**Учебные задачи:**

1. Изучить основные приемы управления дисками и файловыми системами.
2. Отработать на практике основные приемы управления дисками.

**Задачи практического занятия №4**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по управлению дисками.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:

– Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Для того, чтобы не возникало проблем при работе системы и для обеспечения сохранности данных, необходимо проводить различные профилактические и диагностические мероприятия.

**Очистка диска.** При регулярной работе на компьютере иногда накапливается некоторый пользовательский и системный "мусор", который полезно переодически расчищать и ликвидировать. Для этого существует много различных программ, а в Windows существует утилита – Очистка диска. Мусор можно вычищать и вручную. Он обычно хранится в папках с названием Temp и такие файлы имеют расширения tmp.

**Проверка диска.** Файловая система, с которой работает ОС хороша только когда работает без сбоев. Если каким-то образом соответствие между тем, что записано в файловой таблице, и тем, что на самом деле находится на диске, нарушено, последствия могут быть непредсказуемы. Это может возникнуть вследствие сбоев ОС, ПО. В частности, велика вероятность возникновения ошибок при некорректном завершении работы компьютера, при зависании системы и т.д. Обнаружить возникшие проблемы и предотвратить неприятности поможет стандартная программа DOS и Windows Проверка диска или ScanDisk.

**Дефрагментация диска.** Как известно, с точки зрения быстродействия винчестер одно из самых слабых мест системы. К счастью, помогает тот факт, что информацию, которая расположена "подряд", считать можно намного быстрее. Что значит "подряд"? Каждый файл на диске занимает определенное пространство. Это пространство разбито на блоки – кластеры. Каждый кластер принадлежит определенному файлу. Хорошо, если кластеры одного файла идут подряд, но так бывает не всегда.

Файлы на диске постоянно создаются и уничтожаются. Операционная система не всегда может выделить файлу место таким образом, чтобы его кластеры шли друг за другом. То есть файл может занимать несколько кластеров, разбросанных по разным местам диска. В этом случае говорят, что файл фрагментирован. Хотя файл от этого и не портится, но скорость чтения и записи замедляется заметно. Если на диске образуется много таких файлов, то скорость работы системы заметно падает. Для решения этой проблемы помогает программа DOS и Windows Дефрагментация диска или Defrag.

**Архивация данных.**Необходима для создания копий наиболее важных данных, хранящихся на винчестере, для того, чтобы не произошло их потери в случае сбоев системы или отказа работы винчестера. Для этой цели лучше использовать какие-то внешние носители информации – дискеты, компакт-диски, магнитные ленты, другие винчестеры и т.д. Для создания резервных копий данных в DOS и Windows существует программа "Архивация данных" или Backup.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Какие виды приемов для управления и обслуживания дисков существуют?
2. Что такое Проверка диска?
3. Что такое Очистка диска?
4. Что такое Дефрагментация диска?
5. Что такое Архивация данных?

**Задания для практического занятия №4**

1. Провести архивацию данных.
2. Провести анализ логического диска.
3. Провести очистку диска.
4. Провести дефрагментацию диска.

**Инструкция по выполнению заданий практического занятия №4**

1. Внимательно прочитать задание.
2. Выполнить задания строго в соответствии с их порядком.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

Во время выполнения практического задания четко следовать очередности. В выводе обратить внимание на то, чем вызван именно такой порядок выполнения действий

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания в виде скриншотов.
4. Написать вывод о проделанной работе.

## Практическое занятие № 5
## Установка Windows.

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Установка и настройка Windows».

**Учебные задачи:**

1. Познакомиться с основными этапами установки ОС Windows.
2. Отработать на практике процесс установки ОС Windows.

**Образовательные результаты, заявленные во ФГОС СПО**

Студент должен

уметь:

- устанавливать и сопровождать операционные системы;;

знать:

- принципы построения, типы и функции операционных систем;
- машинно-зависимые и машинно-независимые свойства операционных систем.

**Задачи практического занятия №5**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по установке ОС Windows.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   - Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   - Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Операционная система обеспечивает совместное функционирование всех устройств компьютера и предоставляет пользователю доступ к его ресурсам.

Операционные системы разные, но их назначение и функции одинаковые. Операционная система является базовой и необходимой составляющей программного обеспечения компьютера, без нее компьютер не может работать в принципе. В состав современных операционных систем обычно входит следующие основные модули:

- программный модуль, управляющий файловой системой;
- командный процессор, выполняющий команды пользователя;
- драйверы устройств;
- программные модули, обеспечивающие графический пользовательский интерфейс;
- сервисные программы;
- справочная система.

На сегодняшний момент операционная система Windows фирмы Microsoft во всех ее проявлениях бесспорно считается самой распространенной операционной системой на ПК: в мире более 150 млн. IBM PC-совместимых компьютеров, и система Windows установлена на 100 млн. из них. Очевидно, что ознакомление с ПК необходимо начинать с ознакомления с Windows, ведь без нее работа на ПК немыслима для большинства пользователей. Знание системы Windows - необходимый кирпичик в стене познания ПК.

Первоначально, Windows, разрабатывалась не как операционная система, какой мы привыкли видеть современные ее версии, а как графическая оболочка MS-DOS. Надо отметить, что концепция графического интерфейса была разработана отнюдь не Microsoft. Уже за несколько лет до внедрения Windows существовали компьютеры AppleMacintosh, с графической операционной системой (MacOS), интерфейс которой был более дружественным и понятным рядовому пользователю, в отличие от командной строки MS-DOS. Строго говоря, Windows, не единственная попытка избавить пользователя от командной строки на IBM-совместимых компьютерах. Очень известной, в свое время была псевдографическая (на самом деле работавшая в текстовом режиме), оболочка NortonCommander корпорации Symantec. Она ускоряла в несколько раз процесс навигации по дисковому пространству, к тому же, более естественно представляла иерархию каталогов в виде дерева. Однако,Windows появилась раньше Norton, хотя Norton был более популярен, в частности, из - за низких системных требований.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое операционная система?
2. Что включает в себя операционная система?
3. Как происходило развитие ОС Windows?

**Задания для практического занятия №5**

1. Установить ОС Windows.

**Инструкция по выполнению заданий практического занятия №5**

1. Запустить машину на вашем ПК.
2. Скачать с файлового сервера iso-образ ОС Windows.
3. Начать установку ОС Windows.

**Методика анализа результатов, полученных в ходе практического занятия**

В процессе установки ОС Windows будьте предельно внимательны к тем настройкам, которые вы делаете в процессе установки операционной системы

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

**Практическое занятие № 6**

## Изменение параметров загрузки Windows

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Изменение параметров загрузки Windows».

**Учебные задачи:**

1. Изучить этапы и параметры загрузки ОС Windows.
2. Научиться выполнять настройку загрузки ОС Windows.

### Задачи практического занятия №6

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по изменению параметров загрузки ОС.
3. Подготовить отчет по практической работе.

### Обеспеченность занятия (средства обучения)

1. Учебно-методическая литература:
   - Методические указания по выполнению практических занятий.
2. Технические средства обучения:
   - Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

### Краткие теоретические и учебно-методические материалы по теме практического занятия

Файлы операционной системы хранятся во внешней, долговременной памяти (на жестком, гибком или лазерном диске). Однако программы могут выполняться, только если они находятся в оперативной памяти, поэтому файлы операционной системы необходимо загрузить в оперативную память.

Диск (жесткий, гибкий или лазерный), на котором находятся файлы операционной системы и с которого производится ее загрузка, называется системным.

После включения компьютера производится загрузка операционной системы с системного диска в оперативную память. Загрузка должна выполняться в соответствии с программой загрузки. Однако для того чтобы компьютер выполнял какую-нибудь программу, эта программа должна уже находиться в оперативной памяти. Разрешение этого противоречия состоит в последовательной, поэтапной загрузке операционной системы.

**Самотестирование компьютера.** В состав компьютера входит энергонезависимое постоянное запоминающее устройство (ПЗУ), содержащее программы тестирования компьютера и первого этапа загрузки операционной системы - это BIOS.

После включения питания компьютера или нажатия кнопки Reset на системном блоке компьютера или одновременного нажатия комбинации клавиш {Ctrl+Alt+Del} на клавиатуре процессор начинает выполнение программы самотестирования компьютера POST. Производится тестирование работоспособности процессора, памяти и других аппаратных средств компьютера.

В процессе тестирования сначала могут выдаваться диагностические сообщения в виде различных последовательностей коротких и длинных звуковых сигналов. После успешной инициализации видеокарты краткие диагностические сообщения выводятся на экран монитора.

Для установки правильной даты и времени, а также внесения изменений в конфигурацию аппаратных средств компьютера в процессе выполнения самотестирования необходимо нажать клавишу {Del}. Загрузится системная утилита BIOS Setup, имеющая интерфейс в виде системы иерархических меню. Пользователь может установить новые параметры конфигурации компьютера и запомнить их в специальной микросхеме памяти, которая при выключенном компьютере питается от батарейки, установленной на системной плате. В случае выхода из строя батарейки конфигурационные параметры теряются и компьютер перестает нормально загружаться.

**Загрузка операционной системы.** После проведения самотестирования специальная программа, содержащаяся в BIOS, начинает поиск загрузчика операционной системы.

Происходит поочередное обращение к имеющимся в компьютере дискам и поиск на определенном месте наличия специальной программы MasterBoot.

Если системные диски в компьютере отсутствуют, то загрузка операционной системы прекращается и компьютер остается неработоспособным.

После окончания загрузки операционной системы управление передается командному процессору. В случае использования интерфейса командной строки на экране появляется приглашение системы к вводу команд. Приглашение представляет собой последовательность символов, сообщающих о текущем диске и каталоге.

В случае загрузки графического интерфейса операционной системы команды могут вводиться с помощью мыши.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Каковы основные этапы самотестирования компьютера?
2. Что хранится в микросхеме конфигурационной памяти компьютера?
3. Каковы основные этапы загрузки операционной системы?

**Задания для практического занятия №6**

1. Запустить ОС Windows.
2. В процессе запуска ОС вызвать меню параметров загрузки и протестировать каждый вариант загрузки.
3. Проанализировать сходства и различия, а также преимущества и недостатки каждого из предложенных вариантов загрузки.

**Инструкция по выполнению заданий практического занятия №6**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

После выполнения задания, следует еще раз все внимательно проверить на наличие ошибок, особенно это касается настроек параметров загрузки.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 7
## Настройка параметров рабочей среды пользователя в Windows.

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Настройка параметров рабочей среды пользователя Windows».

**Учебные задачи:**

1. Изучить настройку основных компонентов ОС Windows.
2. Научиться выполнять настройку меню, рабочей среды, технических средств.

**Задачи практического занятия №7**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по настройке параметров рабочей среды пользователя.
3. Подготовить отчет по практической работе.

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

**Настройка (или конфигурирование) программного продукта** – это процесс изменения его свойств, выполняемый в целях адаптации программного продукта к техническим средствам ПК, наиболее полного удовлетворения потребностей пользователя, повышения эффективности функционирования программного продукта.

MicrosoftWindows хранит информацию о конфигурации в двух местах: реестре и службе каталогов ActiveDirectory. Модификации реестра или ActiveDirectory приводят к изменению конфигурации Windows. Для изменения реестра или ActiveDirectory используются следующие средства:

- MMC - MicrosoftManagementConsole;
- Панель управления - ControlPanel;
- Редактор реестра - RegistryEditor.

**Конфигурирование операционной системы**

Для конфигурирования операционной системы используются программы в Панели управления. Окно Система позволяет настраивать:

- производительность системы;
- размер реестра;
- переменные среды;
- параметры производительности;
- параметры загрузки и восстановления системы.

Для настройки среды Windows прежде всего просматривает файл Autoexec.bat (если он существует). Затем задаются системные переменные среды. Переменные среды из файла Autoexec.bat и переменные среды пользователя переопределяют системные переменные.

Параметры загрузки и восстановления системы регулируются в окне Система на вкладке Дополнительно. Программы в Панели управления влияют на среду ОС независимо от текущего пользователя системы.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое конфигурирование и настройка ОС?.
2. Какие настройки ОС можно производить?

**Задания для практического занятия №7**

1. Определить настройки системы:
2. Настроить параметры мыши:
3. Настроить параметра экрана

**Инструкция по выполнению заданий практического занятия №7**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Определить настройки системы:
   - установленное оборудование;
   - установленные профили;
   - параметры производительности;
   - параметры загрузки и восстановления системы.
   - переменные среды.
3. Настроить параметры мыши:

- указатель мыши
- параметры указателя;
- колёсико.

4. Настроить параметра экрана:
   - Фоновый рисунок Рабочего стола;
   - Отображаемые на Рабочем столе значки;
   - Оформление Рабочего стола.
5. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

После выполнения задания, следует еще раз все внимательно проверить на наличие ошибок, особенно это касается настроек системы..

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 8
## Создание и администрирование локальной группы в Windows.

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Создание и администрирование локальной группы в Windows».

**Учебные задачи:**

1. Изучить процесс и алгоритм создания новых учетных записей.
2. Отработать на практике создание новых учетных записей
3. Изучить и отработать администрирование локальных групп.

**Задачи практического занятия №8**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по созданию и администрированию учетных записей пользователей.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Учетная запись (аккаунт) – информация, которая заводится для пользователя при его регистрации в какой-либо системе (например, в системе бесплатной электронной почты). У каждого аккаунта есть имя и пароль. Учётная запись содержит сведения, необходимые для идентификации пользователя при подключении к системе, информацию для его авторизации и учёта: имя пользователя и пароль (или другое аналогичное средство аутентификации – например, биометрические характеристики). Пароль (его аналог), в целях его безопасности, необходимо храниться в зашифрованном (хэшированном виде).

При формировании журнала учётных записей в системе фиксируют также информацию о действиях конкретного пользователя в системе: давность последнего входа в систему, продолжительность последнего пребывания в системе, адрес использованного при

подключении компьютера, интенсивность использования системы, суммарное (удельное количество) определённых операций, произведённых в системе и т.д.

Администрирование (административные механизмы) – процедура управления работой компьютерной системы (ПК), регламентирующая некоторые процессы (или их часть), нуждающиеся в целевом управлении и ограничениях. К таким процессам относят: планирование работ, построение, эксплуатацию и поддержку эффективной информационной системы.

«Администрирование» – это также папка в панели управления ПК, содержащая средства для системных администраторов и опытных пользователей (набор средств может различаться в зависимости от версии Windows).

Основной целью системного администрирования (наиболее простого типа администрирования) является приведение сети в соответствие с целями и задачами, для которых эта сеть предназначена. Достигается эта цель путём управления сетью, позволяющего минимизировать затраты времени и ресурсов, направляемых на управление системой, одновременно максимизируя доступность, производительность и продуктивность всей системы.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое учетная запись?
2. Типы учетных записей
3. Что такое авторизация в компьютерной сети?
4. Что такое аутентификация в компьютерной сети?
5. Каковы основные типы учетных записей на локальном ПК и в компьютерной сети?
6. Для чего настраиваются параметры семейной безопасности на локальном ПК?
7. Каковы основные виды ограничений при настройке параметров семейной безопасности?
8. Каковы основные способы разделения ресурсов локального ПК между несколькими пользователями?
9. Для чего формируется журнал учетных записей?
10. Что такое администрирование компьютерной сети?
11. В чем заключается основное отличие службы сетевого администрирования от системного?

**Задания для практического занятия №8**

1. Создать новую учетную запись пользователя ПК.
2. Зайти в ОС ПК под зарегистрированным именем нового пользователя.
3. Отредактировать (изменить) учетную запись пользователя
4. Изучить возможности и настроить параметры семейной безопасности для Вашей учетной записи.
5. Сформировать отчет о последних действиях пользователя ПК.
6. Ознакомиться с задачами администрирования на локальном ПК.
7. Ознакомиться с возможностями администратора ПК по работе с учетными записями пользователей.
8. Ознакомиться с возможностями журнала контроля учетных записей.

**Инструкция по выполнению заданий практического занятия №8**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

В момент выполнения задания, следует внимательно вводить все параметры учетных записей и локальных групп, чтобы избежать ошибок. После выполнения заданий еще раз внимательно все проверить на наличие ошибок, особенно это касается настроек.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 9
## Конфигурирование устройств и установка драйверов устройств в Windows

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Конфигурирование устройств и установка драйверов устройств в Windows».

**Учебные задачи:**

1. Изучить и отработать на практике алгоритм установки драйверов.

### Задачи практического занятия №9

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по установке драйверов.
3. Подготовить отчет по практической работе

### Обеспеченность занятия (средства обучения)

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

### Краткие теоретические и учебно-методические материалы по теме практического занятия

Драйвер - это программный код (специальная программа), функция которой заключается в предоставлении возможности использовать определенную железо (видеокарту, клавиатуру, принтер и т.д.). Так как параметры периферийных устройств меняются от производителя к производителю, то разным пользователям программы может потребоваться дюжина различных драйверов, чтобы он мог работать на имеющемся у него оборудовании.

Причем конкретный драйвер не всегда может быть загружен память. Он туда грузится, лишь по потребности в ресурсах устройства. Так экономится память, но в ущерб скорости.

Имеется четыре способа включения драйверов устройств в программу:

1. Можно поместить код для всех драйверов прямо в программу. Например, чтобы поддерживать различные принтеры, можно создать таблицу управляющих последовательностей и искать в ней нужный код каждый раз, когда он потребуется. Этот подход тратит много памяти и может быть достаточно медленным.
2. Создать ряд драйверов устройств и потребовать, чтобы программа загружала необходимый в качестве оверлея (т.е. помещать его в область программы, специально оставленную для этой цели).
3. Создать драйвер устройства как отдельную программу, которая указывается в командном файле, выполняемом при загрузке системы. Программа запускается и устанавливает драйвер устройства как программу обработки прерывания. После этого программа завершается, но остается резидентной в памяти. Впоследствии наша программа использует этот драйвер через вектор прерывания.

Создать полноценный драйвер устройства, который будет загружаться при старте с помощью файла CONFIG.SYS. MS DOS поддерживает такой тип драйверов устройств и однажды загруженный он может использовать все возможности команд DOS, включая проверку ошибок. Специальная команда IOCTL (Контроль ввода/вывода) позволяет программе узнать статус драйвера и послать ему управляющую строку, помимо обычного потока данных.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое драйвер?
2. Какие способы установки драйверов?

**Задания для практического занятия №9**

1. Запустить Установка нового оборудования и ознакомиться с возможностями и параметрами данной программы..
2. Обновить драйверы на все подключенные периферийные устройства.

**Инструкция по выполнению заданий практического занятия №9**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

Если вдруг драйвер не устанавливается в автоматическом режиме, то действуем по следующему алгоритму:

1. Нажимаем "Win+R".
2. В открывшемся окне набираем "hdwwiz", жмём OK.
3. В открывшемся окне, жмем "Далее" (Next).
4. В следующем окне выбираем "Установка оборудования, выбранного из списка вручную" (Install the hardware that I manually select from a list), жмём "Далее" (Next).
5. Выбираем в списке самую верхнюю строчку "Показать все устройства" (Showalldevices), жмём "Далее" (Next).
6. В следующем окне нажимаем на кнопку "Установить с диска" (InstallFromDisk), в нижнем правом углу.
7. В открывшемся окне нажимаем кнопку "Обзор" (Browse) и указываем путь к файлу драйвера (inf-файл), нажимаем Ok, а затем "Далее" (Next).
8. Подтверждаем желание установить нажатием "Далее" (Next).

**Методика анализа результатов, полученных в ходе практического занятия**

При скачивании с Интернета драйверов под подключенные периферийные устройства, необходимо внимательно проверять модель устройства на вашем ПК на совместимость с выбранным вами драйвером.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 10
## Настройка сетевых подключений.

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Настройка сетевых подключений».

**Учебные задачи:**

1. Изучить основные этапы настройки сетевого адаптера.
2. Научиться настраивать LAN

**Задачи практического занятия №10**

1. Повторить краткие теоретические сведения по теме практического задания

2. Выполнить работу по настройке сетевых подключений.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Windows, MSWord.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

**Сетевая карта или сетевой** адаптер - это плата расширения, вставляемая в разъем материнской платы (main board) компьютера. Также существуют сетевые адаптеры стандарта PCMCIA для ноутбуков (notebook), они вставляются в специальный разъем в корпусе ноутбука. Или интегрированные на материнской плате компьютера, они подключаются по какой либо локальной шине. Появились Ethernet сетевые карты, подключаемые к USB (Universal Serial Bus) порту компьютера.

Для определения точки назначения пакетов (frames) в сети Ethernet используется **MAC-адрес**. Это уникальный серийный номер присваиваемый каждому сетевому устройству Ethernet для идентификации его в сети. MAC-адрес присваивается адаптеру его производителем, но может быть изменен с помощью программы. Делать это не рекомендуется ( только в случае обнаружения двух устройств в сети с одним MAC- адресом). При работе сетевые адаптеры просматривают весь проходящий сетевой трафик и ищут в каждом пакете свой MAC-адрес. Если таковой находится, то устройство (адаптер) декодирует этот пакет. Существуют также специальные способы по рассылке пакетов всем устройствам сети одновременно (broadcasting). MAC-адрес имеет длину 6 байт и обычно записывается в шестнадцатиричном виде, например 12:34:56:78:90:AB.

**Конфигурирование сетевой платы**

Для нормальной работы каждой сетевой платы ей необходимы адрес ввода-вывода (In/Out port) и номер прерывания (IrQ).

Конфигурирование сетевой платы заключается в настройке ее на свободные адрес и прерывание, которые затем будут использоваться операционной системой. Адрес (i/o port) и прерывание(IrQ) для каждой сетевой платы должно быть свое, отличное от других устройств компьютера. Современные сетевые карты, поддерживающие технологию Plug-n-play сами выполняют эту операцию, для всех остальных необходимо самим проделать ее.

Порядок настройки сетевой карты зависит от ее модели и конфигурации. Большинство современных сетевых карт поддерживают стандарт Plug-And- Play, и операционная система автоматически обнаруживает эти устройства после их установки и включения питания компьютера: при этом пользователю достаточно лишь указать в соответствующем окне, откуда система должна копировать соответствующие драйверы. Более старые сетевые адаптеры (в основном, подключаемые к шине ISA) не определяются в Windows автоматически и требуют настройки вручную.

В ряде случаев на самой плате сетевого адаптера имеется набор перемычек или переключателей, посредством которых можно выставить режим его настройки.

Внимательно ознакомьтесь с технической документацией вашей сетевой карты прежде, чем приступить к ее конфигурированию. Однако, в некоторых случаях автоматическая настройка Plug-And-Play-адаптеров происходит некорректно, и в результате возникают аппаратные конфликты между сетевой картой и иным оборудованием. Как правило, подобные ситуации бывают вызваны тем, что несколько различных устройств начинают несанкционированно использовать одни и те же ресурсы, например запрос на прерывание (IRQ, Interrupt Request), адреса каналов непосредственного доступа к памяти (DMA, Direct Memory Access) или диапазон ввода-вывода (I/O Range).

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое сетевой адаптер?.
2. Характеристикисетевогоадаптера.
3. Что надо учитывать при выборе сетевого адаптера?
4. Виды сетевых адаптеров.
5. Конфигурирование сетевого адаптера.
6. Назначение папки "Мое сетевое окружение".

**Задания для практического занятия №10**

1. Установить сетевой адаптер.
2. Выполнить его настройку.
3. Создать новое сетевое подключение.

**Инструкция по выполнению заданий практического занятия №10**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

В момент выполнения задания, следует внимательно вводить все параметры настроек, чтобы избежать ошибок.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

**Практическое занятие № 11**
**Управление дисками и файловыми системами в Linux**

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Управление дисками и файловыми системами в Linux».

**Учебные задачи:**

1. Изучить структуры файловой системы ОС LINUX,
2. Изучить команды создания, удаления, модификации файлов и каталогов, функций манипулирования данными.

**Задачи практического занятия №11**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить задание по работе в файловой системе.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Linux, OpenOffice.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

В операционной системе LINUX файлами считаются обычные файлы, каталоги, а также специальные файлы, соответствующие периферийным устройствам (каждое устройство представляется в виде файла). Доступ ко всем файлам однотипный, в том числе, и к файлам периферийных устройств. Такой подход обеспечивает независимость программы пользователя от особенностей ввода/вывода на конкретное внешнее устройство.

Файловая структура LINUX имеет иерархическую древовидную структуру. В корневом каталоге размещаются другие каталоги и файлы, включая 5 основных каталогов:

bin - большинство выполняемых командных программ и shell - процедур;

tmp - временные файлы;

usr - каталоги пользователей (условное обозначение);

etc - преимущественно административные утилиты и файлы;

dev - специальные файлы, представляющие периферийные устройства; при добавлении периферийного устройства в каталог /dev должен быть добавлен соответствующий файл (черта / означает принадлежность корневому каталогу).

Текущий каталог - это каталог, в котором в данный момент находится пользователь. При наличии прав доступа, пользователь может перейти после входа в систему в другой каталог. Текущий каталог обозначается точкой (.); родительский каталог, которому принадлежит текущий, обозначается двумя точками (..).

Полное имя файла может включать имена каталогов, включая корневой, разделенных косой чертой, например: /home/student/file.txt. Первая косая черта обозначает корневой каталог, и поиск файла будет начинаться с него, а затем в каталоге home, затем в каталоге student.

Один файл можно сделать принадлежащим нескольким каталогам. Для этого используется команда ln (link).

В LINUX различаются 3 уровня доступа к файлам и каталогам:

1) доступ владельца файла;

2) доступ группы пользователей, к которой принадлежит владелец файла;

3) остальные пользователи.

Для каждого уровня существуют свои байты атрибутов, значение которых расшифровывается следующим образом:

r – разрешение на чтение;

w – разрешение на запись;

x – разрешение на выполнение;

- – отсутствие разрешения.

В домашнем каталоге пользователь имеет полный доступ к файлам (READ, WRITE, EXECUTE; r, w, x).

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что считается файлами в ОС LINUX?
2. Объясните назначение связей с файлами и способы их создания.
3. Что определяет атрибуты файлов и каким образом их можно просмотреть и изменить?
4. Какие методы создания и удаления файлов, каталогов Вы знаете?
5. В чем заключается поиск по шаблону?
6. Какой командой можно получить список работающих пользователей и сохранить его в файле?

**Задания для практического занятия №11**

1. Ознакомиться с файловой структурой ОС LINUX. Изучить команды работы с файлами.
2. Используя команды ОС LINUX, создать два текстовых файла.
3. Полученные файлы объединить в один файл и его содержимое просмотреть на экране.
4. Создать новую директорию и переместить в нее полученные файлы.
5. Вывести полную информацию обо всех файлах и проанализировать уровни доступа.
6. Добавить для всех трех файлов право выполнения членам группы и остальным пользователям.
7. Просмотретьатрибутыфайлов.
8. Создатьещеодинкаталог.
9. Установить дополнительную связь объединенного файла с новым каталогом, но под другим именем.
10. Сделать текущим новый каталог и вывести на экран расширенный список информации о его файлах.
11. Произвести поиск заданной последовательности символов в файлах текущей директории и получить перечень соответствующих файлов.
12. Получить информацию об активных процессах и имена других пользователей.

**Инструкция по выполнению заданий практического занятия №11**

1. Внимательно прочитать задание.

2. При работе в командной строке внимательно вводите команды и названия файлов и папок.
3. При написании отчета не забудьте вставить скриншоты вашей работы.

**Методика анализа результатов, полученных в ходе практического занятия**

После выполнения задания, следует еще раз все внимательно проверить на наличие ошибок, особенно это касается команд и написания названий файлов и папок.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 12
## Установка Linux

**Учебная цель:**

Приобрести опыт установки операционной системы Linux.

**Учебные задачи:**

1. Установить ОС на виртуальный компьютер.
2. Разобрать процесс установки ОС на этапы.
3. Познакомиться с основными группами программ входящих в состав ОС.

**Задачи практического занятия №12**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить установку ОС Linux.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Linux, OpenOffice.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Linux (полное название GNU/Linux) — общее название UNIX-подобных операционных систем на основе одноимённого ядра и собранных для него библиотек и системных программ, разработанных в рамках проекта GNU.

GNU/Linux работает на PC-совместимых системах семейства Intel x86, а также на IA64, AMD64, PowerPC, ARM и многих других.

К операционной системе GNU/Linux также часто относят программы, дополняющие эту операционную систему, и прикладные программы, делающие её полноценной многофункциональной операционной средой.

В отличие от большинства других операционных систем, GNU/Linux не имеет единой «официальной» комплектации. Вместо этого GNU/Linux поставляется в большом количестве так называемых дистрибутивов, в которых программы GNU соединяются с ядром Linux и другими программами.

Дистрибутив — это не просто набор программ, а ряд решений для разных задач пользователей, объединённых едиными системами установки, управления и обновления пакетов, настройки и поддержки.

Самые распространённые в мире дистрибутивы:

•Ubuntu — быстро завоевавший популярность дистрибутив, ориентированный на лёгкость в освоении и использовании.

•openSUSE — бесплатно распространяемая версия дистрибутива SuSE, принадлежащая компании Novell. Отличается удобством в настройке и обслуживании благодаря использованию утилиты YaST.

•Fedora — поддерживается сообществом и корпорацией RedHat, предшествует выпускам коммерческой версии RHEL.

•Debian GNU/Linux — международный дистрибутив, разрабатываемый обширным сообществом разработчиков в некоммерческих целях. Послужил основой для создания множества других дистрибутивов. Отличается строгим подходом к включению несвободного ПО.

•Archlinux — ориентированный на применение самых последних версий программ и постоянно обновляемый, поддерживающий одинаково как бинарную, так и установку из исходных кодов и построенный на философии простоты KISS, этот дистрибутив ориентирован на компетентных пользователей, которые хотят иметь всю силу и модифицируемость Linux, но не в жертву времени обслуживания.

Помимо перечисленных, существует множество других дистрибутивов, как базирующихся на перечисленных, так и созданных с нуля и зачастую предназначенных для выполнения ограниченного количества задач.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое операционная система?
2. Что включает в себя операционная система?
3. Как происходило развитие ОС Linux?

**Задания для практического занятия №12**

1. Установить ОС Linux.

**Инструкция по выполнению заданий практического занятия №12**

1. Запустить машину на вашем ПК.
2. Скачать с файлового сервера iso-образ ОС Linux.
3. Начать установку ОС Linux.

**Методика анализа результатов, полученных в ходе практического занятия**

В процессе установки ОС Linux будьте предельно внимательны к тем настройкам, которые вы делаете в процессе установки операционной системы

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 13
## Создание и изменение параметров учетных записей в Linux

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Создание и изменение параметров учетных записей в Linux».

**Учебные задачи:**

1. Изучить процесс и алгоритм создания новых учетных записей.
2. Отработать на практике создание новых учетных записей

3. Изучить и отработать администрирование локальных групп.

**Задачи практического занятия №13**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по созданию и администрированию учетных записей пользователей.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Linux, OpenOffice.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Linux — система многопользовательская, а потому пользователь — ключевое понятие для организации всей системы доступа в Linux. Когда пользователь регистрируется в системе (проходит процедуру авторизации, например, вводя системное имя и пароль), он идентифицируется с **учётной записью**, в которой система хранит информацию о каждом пользователе: его системное имя и некоторые другие сведения, необходимые для работы с ним. Именно с учётными записями, а не с самими пользователями, и работает система. Ниже приведён список этих сведений.

**Системное имя (username)** - это то имя, которое вводит пользователь в ответ на приглашение login. Оно может содержать только латинские буквы и знак "_". Это имя используется также в качестве имени учётной записи.

Linux связывает системное имя с **идентификатором пользователя** в системе — **UID (User ID).** UID — это положительное целое число, по которому система и отслеживает пользователей. Обычно это число выбирается автоматически при регистрации учётной записи, однако оно не может быть совершенно произвольным. В Linux есть некоторые соглашения относительно того, каким типам пользователей могут быть выданы идентификаторы из того или иного диапазона. В частности, UID от "0" до "100" зарезервированы для псевдопользователей.

Кроме идентификационного номера пользователя с учётной записью связан **идентификатор группы.** Группы пользователей применяются для организации доступа нескольких пользователей к некоторым ресурсам. У группы, так же, как и у пользователя, есть имя и идентификационный номер — GID (Group ID). В Linux каждый пользователь должен принадлежать как минимум к одной группе — группе по умолчанию.

При создании учётной записи пользователя обычно создаётся и группа, имя которой совпадает с системным именем, именно эта группа будет использоваться как группа по умолчанию для этого пользователя. Пользователь может входить более чем в одну группу, но в учётной записи указывается только номер группы по умолчанию. Группы позволяют регулировать доступ нескольких пользователей к различным ресурсам.

Помимо системного имени в учётной записи содержится и **полное имя** (имя и фамилия) использующего данную учётную запись человека. Конечно, пользователь может указать что угодно в качестве своего имени и фамилии. Полное имя необходимо не столько системе, сколько людям — чтобы иметь возможность определить, кому принадлежит учётная запись.

Файлы всех пользователей в Linux хранятся раздельно, у каждого пользователя есть собственный **домашний каталог**, в котором он может хранить свои данные. Доступ других

пользователей к домашнему каталогу пользователя может быть ограничен. Информация о домашнем каталоге обязательно должна присутствовать в учётной записи, потому что именно с него начинает работу пользователь, зарегистрировавшийся в системе.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое системное имя?
2. Что такое идентификатор пользователя?
3. Что такое идентификатор группы?
4. Что такое полное имя?
5. Что такое домашний каталог?

**Задания для практического занятия №13**

1. Войдите в систему.
2. Используя Центр управления, создайте своего пользователя.
3. Перейдите в первую текстовую консоль и зарегистрируйтесь пользователем root. Просмотрите файл /etc/passwd. Убедитесь, что созданный вами пользователь появился.
4. Создайте другого пользователя.
5. Задайтесозданномупользователюпароль.
6. Просмотрите файл /etc/passwd.
7. Средствами текстового редактора vi введите в файл /etc/passwd информацию о новом пользователе. Проверьте правильность его создания средствами Центра управления.
8. Просмотрите содержимое домашних каталогов пользователей. Создайте с помощью редактора vi какой–либо текстовый файл в домашнем каталоге пользователя.
9. Создайте новые проектные группы всеми известными вам методами (окно Центр управления, команда groupadd).

**Инструкция по выполнению заданий практического занятия №13**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

В момент выполнения задания, следует внимательно вводить все параметры учетных записей и локальных групп, чтобы избежать ошибок. После выполнения заданий еще раз внимательно все проверить на наличие ошибок, особенно это касается настроек.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

**Практическое занятие № 14**
**Изменение параметров рабочей среды Linux**

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Изменение параметров рабочей среды Linux».

**Учебные задачи:**

1. Изучить различные графические среды.

**Задачи практического занятия №14**

1. Повторить краткие теоретические сведения по теме практического задания

2. Выполнить работу с 2 видами интерфейса.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   - Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   - Персональный компьютер
3. Программное обеспечение: ОС Linux, OpenOffice.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

В Linux существует три основных среды рабочего стола или еще говорят — графического окружения. Это означает, что выбрав дистрибутив вы еще должны выбрать ту графическую среду, которая вам нравится больше и скачивать дистрибутив именно с этим графическим окружением.

Итак, графическая среда рабочего стола определяет внешний вид операционной системы и именно поэтому существует огромное многообразие Linux отличающихся внешне и порой весьма значительно.

Существуют три основные среды рабочего стола — GNOME, KDE и Xfce.

Каждая среда — это отдельный проект, развивающийся самостоятельно и, как принято в Linux, очень часто из проекта рождаются подпроекты, базирующиеся на родителе, но отличающиеся от него и порой очень сильно.

У разных дистрибутивов Linux могут быть доступны разные среды рабочего стола или их модификации. Например, для LinuxMint на данный момент официально существует четыре среды рабочего стола, которые представлены на официальном сайте. Это означает, что вроде бы операционная система одна — LinuxMint, но установив четыре ее модификации вы обнаружите что их интерфейс, то есть внешний вид, отличается.

Это многообразие графических окружений так велико, что дать рекомендации по выбору, то есть какая из сред лучше, просто невозможно. К тому же внешний вид — это весьма субъективная вещь и у каждого пользователя ощущения удобства или «красивости» свои. Поэтому вы сами должны сделать выбор для себя

И еще один нюанс — от установленного графического окружения будет зависеть не только внешний вид операционной системы, но и ее настройки, содержания контекстных меню и расположение различных элементов управления. Причем отличия эти порой весьма существенны.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое среда рабочего стола?.
2. Какие основные графические среды существуют?

**Задания для практического занятия №14**

1. Установить среду рабочего стола Kali
2. Поменять среду рабочего стола на Cinnamon
3. Поменять среду рабочего стола на Xfce
4. Поменять среду рабочего стола на KDE
5. Поменять среду рабочего стола на GNOME
6. Поменять среду рабочего стола на LXDE
7. Поменять среду рабочего стола на MATE

**Инструкция по выполнению заданий практического занятия №14**

1. Поочередно установить каждую графическую среду и ознакомиться с ее внешним видом и возможностями администрирования системы.
2. Сравнить и проанализировать полученные результаты.

**Методика анализа результатов, полученных в ходе практического занятия**

В отчете кратко описать графическую среду и приложить скриншоты работы с ней.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## Практическое занятие № 15
## Подключение и конфигурирование аппаратных устройств в Linux.

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Подключение и конфигурирование аппаратных устройств в Linux».

**Учебные задачи:**

1. Изучить аппаратные устройства.
2. Научиться подключать и конфигурировать аппаратные устройства в ОС Linux.

**Задачи практического занятия №15**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу с 2 видами интерфейса.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   - Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   - Персональный компьютер
3. Программное обеспечение: ОС Linux, OpenOffice.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Одной из основных задач операционной системы является управление аппаратной частью. Ту программу или тот кусок программного кода, который предназначен для управления конкретным устройством, и называют обычно драйвером устройства. Необходимость драйверов устройств в операционной системе объясняется тем, что каждое отдельное устройство воспринимает только свой строго фиксированный набор специализированных команд, с помощью которых этим устройством можно управлять. Причем команды эти чаще всего предназначены для выполнения каких-то простых элементарных операций. Если бы каждое приложение вынуждено было использовать только эти команды, писать приложения было бы очень сложно, да и размер их был бы очень велик. Поэтому приложения обычно используют какие-то команды высокого уровня (типа "записать файл на диск"), а о преобразовании этих команд в управляющие последовательности для конкретного устройства заботится драйвер этого устройства. Поэтому каждое отдельное устройство, будь то дисковод, клавиатура или принтер, должно иметь свой программный драйвер, который выполняет роль транслятора или связующего звена между аппаратной частью устройства и программными приложениями, использующими это устройство.

В Linux драйверы устройств бывают трех типов:

Драйверы первого типа являются частью программного кода ядра (встроены в ядро). Соответствующие устройства автоматически обнаруживаются системой и становятся доступны для приложений. Обычно таким образом обеспечивается поддержка тех устройств, которые необходимы для монтирования корневой файловой системы и запуска компьютера. Примерами

таких устройств являются стандартный видеоконтроллер VGA, контроллеры IDE-дисков, материнская плата, последовательные и параллельные порты.

Драйверы второго типа представлены модулями ядра. Они оформлены в виде отдельных файлов и для их подключения (на этапе загрузки или впоследствии) необходимо выполнить отдельную команду подключения модуля, после чего будет обеспечено управление соответствующим устройством. Если необходимость в использовании устройства отпала, модуль можно выгрузить из памяти (отключить). Поэтому использование модулей обеспечивает большую гибкость, так как каждый такой драйвер может быть переконфигурирован без остановки системы. Модули часто используются для управления такими устройствами как SCSI-адаптеры, звуковые и сетевые карты.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Что такое аппаратные устройства?
2. Какие типы драйверов бывают в ОС Linux?

**Задания для практического занятия №15**

1. Установить утилиту для автоматического подключения новых устройств.
2. Создать собственную раскладку клавиатуры.
3. Установить модуль XKB.
4. Настроить звуковую карту.

**Инструкция по выполнению заданий практического занятия №15**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

При скачивании с Интернета драйверов под аппаратные устройства, необходимо внимательно проверять модель устройства на вашем ПК на совместимость с выбранным вами драйвером.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

**Практическое занятие № 16**
**Поддержка приложений Windows.**

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Поддержка приложений Windows».

**Учебные задачи:**

1. Изучить поддержку приложений Windows.
2. Отработать на практике работу приложений Windows в ОС Linux.

**Задачи практического занятия №16**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу с 2 видами интерфейса.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   – Методические указания по выполнению практических занятий;.

2. Технические средства обучения:
   – Персональный компьютер
3. Программное обеспечение: ОС Linux, OpenOffice.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Один из популярных способов работы с Windows-приложениями в Linux – использование виртуальных машин. Как и у всех, у него есть свои достоинства и недостатки.

К первым, помимо простоты, можно отнести возможность запуска приложений в их «родной» среде, здесь стоит сделать оговорку. MicrosoftWindows в данном случае будет работать как обычное приложение — в отдельном окне и будет называться «гостевой ОС», в то время как основная ОС называется «хост-система» или «хост-ОС», данный подход повышает стабильность работы самого приложения. Приложения, выполняемые в этом случае в гостевой ОС будут изолированы от основной ОС, и если вдруг гостевая ОС будет заражена сетевым червем или произойдет неисправимый сбой, на хост-ОС это никак не повлияет.

К плюсам относится возможность работать со «снимками» системы, то есть делать запись текущего состояния ОС и при необходимости (вирус заразил всю систему) восстановить предыдущее состояние, а также, созданный мастером образ гостевой операционной системы можно переносить на любой компьютер и другое.

К основным недостаткам — необходимость запуска целой операционной системы, что вызывает снижение производительности основной ОС (так как виртуальная машина использует достаточно много системных ресурсов), долгое время запуска самого приложения (от запуска эмулятора до момента открытия приложения) и, как бы противоречиво это не звучало изолированность системы, которая ограничивает возможности по обмену данными с хост-системой.

Суть метода сводится к тому, что в программе эмуляторе в оконном режиме запускается одна из версий операционной системы Windows. В этом окне вы работаете с гостевой ОС как с обычной, устанавливаете нужные программы и запускаете их. Получится, что в одном окне работает операционная система в которой работает нужное приложение.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Способы работы с Windows-приложениями в Linux.
2. Что такое виртуальная машина?

**Задания для практического занятия №16**

1. Установить виртуальную машину.
2. Установить на виртуальную машину ОС Window.
3. Установить Wine.

**Инструкция по выполнению заданий практического занятия №16**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

При скачивании с Интернета драйверов и программ, необходимо внимательно проверять модель устройства на вашем ПК на совместимость с выбранным вами драйвером.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

**Практическое занятие № 17**
**Настройка сетевых подключений в Linux.**

**Учебная цель:**

Закрепление полученных теоретических знаний по теме «Настройка сетевых подключений в Linux.».

**Учебные задачи:**

1. Изучить настройку сетевых подключений.
2. Отработать на практике настройку сетевых подключений.

**Задачи практического занятия №17**

1. Повторить краткие теоретические сведения по теме практического задания
2. Выполнить работу по настройке сетевых подключений.
3. Подготовить отчет по практической работе

**Обеспеченность занятия (средства обучения)**

1. Учебно-методическая литература:
   - Методические указания по выполнению практических занятий;.
2. Технические средства обучения:
   - Персональный компьютер
3. Программное обеспечение: ОС Linux, OpenOffice.

**Краткие теоретические и учебно-методические материалы по теме практического занятия**

Для работы с сетевыми протоколами TCP/IP в Linux достаточно наличие только **петлевого интерфейса**, но если необходимо объединить хосты между собой, естественно, необходимо наличие сетевого интерфейса, каналов передачи данных (например витая пара), возможно, какого-либо сетевого оборудования. Так же, необходимо наличие установленных утилит для настройки сети(/sbin/ifconfig, /sbin/route и др.), обычно поставляемые в пакете net-tools. Так же необходимо наличие конфигурационных файлов для сети (например /etc/hosts) и поддержку сети ядром Linux.

Начнем понимание сетевых механизмов Linux с ручного конфигурирования сети, то есть со случая, когда **IP адрес** сетевого интерфейса **статичен**. Итак, при настройке сети, необходимо учесть и настроить следующие параметры:

**IP-адрес** - это уникальный адрес машины, в формате четырех десятичных чисел, разделенных точками. Обычно, при работе в локальной сети, выбирается из частных диапазонов, например: 192.168.0.1

**Маска подсети** - так же, 4 десятичных числа, определяющие, какая часть адреса относиться к адресу сети/подсети, а какая к адресу хоста. Маска подсети является числом, которое складывается (в двоичной форме) при помощи логического И, с IP-адресом и в результате чего выясняется, к какой подсети принадлежит адрес. Например адрес 192.168.0.2 с маской 255.255.255.0 принадлежит подсети 192.168.0.

**Адрес подсети** - определяется маской подсети. При этом, для петлевых интерфейсов не существует подсетей.

**Широковещательный адрес** - адрес, используемый для отправки широковещательных пакетов, которые получат все хосты подсети. Обычно, он равен адресу подсети со значением хоста 255, то есть для подсети 192.168.0 широковещательным будет 192.168.0.255, аналогично, для подсети 192.168 широковещательным будет 192.168.255.255. Для петлевых интерфейсов не существует широковещательного адреса.

**IP адрес шлюза -** это адрес машины, являющейся шлюзом по-умолчанию для связи с внешним миром. Шлюзов может быть несколько, если компьютер подключен к нескольким сетям одновременно. Адрес шлюза не используется в изолированных сетях (не подключенных к глобальной сети), потому что данным сетям некуда отправлять пакеты вне сети, то же самое относиться и к петлевым интерфейсам.

**IP-адрес сервера имен (DNS - сервера)** - адрес сервера преобразующего имена хостов в IP адреса. Обычно, предоставляется провайдером.

**Вопросы для закрепления теоретического материала к практическому занятию**

1. Какие логические части должен иметь IP-адрес?
2. Для чего необходимо использование маски сети?
3. Что такое классы адресов? Какие классы IP-адресов существуют, чем они характеризуются? К какому классу принадлежат адреса, использованные в работе?
4. Для чего используются индивидуальные и групповые адреса?

**Задания для практического занятия №17**

1. Настроить подключение по локальной сети вручную.
2. Проверить возможность пересылки пакетов.
3. Просмотр сведений о доступных подключениях.
4. Просмотр информации о сети.
5. Измерение скорости Интернет-соединения.
6. Определение маршрута следования пакетов (трассировка).

**Инструкция по выполнению заданий практического занятия №17**

1. Внимательно прочитать задание и краткие теоретические сведения.
2. Выполнить задание.
3. Оформить отчет по практической работе.

**Методика анализа результатов, полученных в ходе практического занятия**

После выполнения задания, следует еще раз все внимательно проверить на наличие ошибок, особенно это касается настроек параметров.

**Порядок выполнения отчета по практическому занятию**

1. Обязательно указать цели и задачи практического занятия.
2. Выписать задание
3. Показать этапы и результат выполнения задания.
4. Написать вывод о проделанной работе.

## 2.3. Перечень вопросов для подготовки обучающихся к промежуточной аттестации

### 1. Функции ОС

ОС – это комплекс взаимосвязанных программ, который действует как интерфейс между приложениями и пользователями с одной стороны и аппаратной частью 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. 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. 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. Операционная система не только предоставляет пользователям и программистам удобный интерфейс к аппаратным средствам компьютера, но и является механизмом, распределяющим ресурсы компьютера. Управление ресурсами включает решение следующих общих, не зависящих от типа ресурса задач: 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. Таким образом, управление ресурсами составляет важную часть функций любой операционной системы, в особенности мультипрограммной. В отличие от функций расширенной машины большинство функций управления ресурсами выполняются операционной системой автоматически и прикладному программисту недоступны.

### 2. Примеры ОС

Операционные системы, развиваясь вместе с ЭВМ, прошли длинный путь от простейших программ в машинных кодах длинной в несколько килобайт до программ, написанных на языках высокого уровня, размер которых исчисляется десятками мегабайт. Такой значительный рост размера операционных систем обусловлен, главным образом, стремлением разработчиков ’украсить’ операционную систему, расширить ее возможности, добавить возможности, изначально несвойственные операционным системам, а также сделать интерфейс пользователя интуитивным. Все эти попытки дали свои результаты, и положительные, и отрицательные. На сегодняшний день на рынке программного обеспечения для IBM PC-совместимых компьютеров сосуществуют несколько семейств операционных систем. Однозадачные однопользовательские ОС MS-DOS и PC-DOS (первая версия этой ОС, выпущенной корпорацией Microsoft в 1981, была предназначена для поставки с компьютерами IBM PC) являются самыми распространенными ввиду своей простоты и ’неприхотливости’, большую роль здесь играет и то, что подавляющее большинство программ работает именно под их управлением. MS-DOS и PC-DOS характеризуются минимальным пользовательским и программным интерфейсами, в тоже время, работая со всевозможными программными оболочками, интегрированными средами (такими как MicrosoftWindows или DESQview), создают комфортабельную среду для пользователя и программы. ОС MicrosoftWindows NT (32-разрядная Windows NT, первая

версия, которой появилась на рынке в 1993-м, а последняя - в 1998 году, с самого начала создавалась как сверхстабильная, надежная система, рассчитанная, прежде всего на работу, а не на разные игрушки-развлечения.), ориентированная на работу в разнородных сетях, высоконадежна, однако, это достигнуто за счет частичной потери совместимости с MS-DOS.Операционная система OS/2 (нестабильность Windows не была секретом ни для кого, в том числе и для разработчиков Microsoft, поэтому параллельно с совершенствованием Windows корпорация вела активную работу по созданию более совершенной и защищенной системы - OS/2) стоит особняком: будучи полноправной многозадачной операционной системой со своим оригинальным графическим пользовательским и программным интерфейсами, она сохраняет совместимость с MS-DOS и PC-DOS (начиная с версии WARP 3.0 и с MicrosoftWindows). ОС UNIX - одна из старейших и наиболее простых операционных систем, изначально была рассчитана на разработку программ (для нее самой и не только) на мини-ЭВМ и позволяла без больших затрат труда программиста переносить программу из одной системы ЭВМ на другую. Неудивительно, что сейчас продается много различных вариантов мобильной операционной системы UNIX, таких как XENIX, UNIXWARE, SUN-OS, LINUX, BSD.

### 3. Эволюция ОС.

История ОС насчитывает примерно полвека. Она во многом определялась и определяется развитием элементной базы и вычислительной аппаратуры. Первые цифровые вычислительные машины, появившиеся в начале 40-х годов, работали без операционных систем, все задачи организации вычислительного процесса решались вручную каждым программистом с пульта управления. Прообразом современных операционных систем явились мониторные системы середины 50-х, которые автоматизировали действия оператора по выполнению пакета заданий. Был создан один из первых языков программирования – Фортран (FormulaTranslation). Для решения экономических задач был создан язык программирования - Кобол. В 1965-1975 годах переход к интегральным микросхемам открыл путь к появлению следующего поколения компьютеров, ярким представителем которых является IBM/360. В этот период были реализованы практически все основные концепции, присущие современным ОС: мультипрограммирование, мультипроцессирование, многотерминальный режим, виртуальная память, файловые системы, разграничение доступа и сетевая работа. В конце 60-х были начаты работы по созданию глобальной сети ARPANET, явившейся отправной точкой для Интернета. К середине 70-х годов широкое распространение получили мини-компьютеры. Их экономичность и доступность послужила мощным стимулом для создания локальных сетей. С середины 70-х годов началось массовое использование UNIX, уникальной для того времени ОС, которая сравнительно легко переносилась на различные типы компьютеров. Хотя ОС UNIX была первоначально разработана для мини-компьютеров, ее гибкость, элегантность, мощные функциональные возможности и открытость позволили ей занять прочные позиции во всех классах компьютеров. В конце 70-х годов был создан рабочий вариант стека протоколов TCP/IP. В 1983 году стек протоколов TCP/IP был стандартизован. Независимость от производителей, гибкость и эффективность, доказанные успешной работой в Интернете, сделали протоколы TCP/IP не только главным транспортным механизмом Интернета, но и основным стеком большинства сетевых ОС. Начало 80-х годов связано со знаменательным для истории операционных систем событием - появлением персональных компьютеров, которые послужили мощным катализатором для бурного роста локальных сетей, создав для этого отличную материальную основу в виде десятков и сотен компьютеров, расположенных в пределах одного здания. В 80-е годы были приняты основные стандарты на коммуникационные технологии для локальных сетей: в 1980 году - Ethernet, в 1985 - TokenRing, в конце 80-х - FDDI. Это позволило обеспечить совместимость сетевых ОС на нижних уровнях, а также стандартизовать интерфейс ОС с драйверами сетевых адаптеров. К началу 90-х практически все

ОС стали сетевыми, способными поддерживать работу с разнородными клиентами и серверами. Появились специализированные сетевые ОС, предназначенные исключительно для выполнения коммуникационных задач, например система IOS компании CiscoSystems, работающая в маршрутизаторах. Особое внимание в течение всего последнего десятилетия уделялось корпоративным сетевым ОС, для которых характерны высокая степень масштабируемости, поддержка сетевой работы, развитые средства обеспечения безопасности, способность работать в гетерогенной среде, наличие средств централизованного администрирования и управления.

4. **Основные принципы построения ОС.**

Простейшая структуризация ОС состоит в разделении всех компонентов ОС на модули, выполняющие основные функции ОС (ядро), и модули, выполняющие вспомогательные функции ОС. Вспомогательные модули ОС оформляются либо в виде приложений (утилиты и системные обрабатывающие программы), либо в виде библиотек процедур. Вспомогательные модули загружаются в оперативную память только на время выполнения своих функций, то есть являются транзитными. Модули ядра постоянно находятся в оперативной памяти, то есть являются резидентными. При наличии аппаратной поддержки режимов с разными уровнями полномочий устойчивость ОС может быть повышена путем выполнения функций ядра в привилегированном режиме, а вспомогательных модулей ОС и приложений - в пользовательском. Это дает возможность защитить коды и данные ОС и приложений от несанкционированного доступа. ОС может выступать в роли арбитра в спорах приложений за ресурсы. Ядро, являясь структурным элементом ОС, в свою очередь, может быть логически разложено на следующие слои (начиная с самого нижнего): машинно-зависимые компоненты ОС; базовые механизмы ядра; менеджеры ресурсов; интерфейс системных вызовов. В многослойной системе каждый слой обслуживает вышележащий слой, выполняя для него некоторый набор функций, которые образуют межслойный интерфейс. На основе функций нижележащего слоя следующий вверх по иерархии слой строит свои функции - более сложные и более мощные, которые, в свою очередь, оказываются примитивами для создания еще более мощных функций вышележащего слоя. Многослойная организация ОС существенно упрощает разработку и модернизацию системы. Микроядерная архитектура является альтернативой классическому способу построения операционной системы, в соответствии с которым все основные функции операционной системы, составляющие многослойное ядро, выполняются в привилегированном режиме. В микроядерных ОС в привилегированном режиме остается работать только очень небольшая часть ОС, называемая микроядром. Все остальные высокоуровневые функции ядра оформляются в виде приложений, работающих в пользовательском режиме. Микроядерные ОС удовлетворяют большинству требований, предъявляемых к современным ОС, обладая переносимостью, расширяемостью, надежностью и создавая хорошие предпосылки для поддержки распределенных приложений. За эти достоинства приходится платить снижением производительности, что является основным недостатком микроядерной архитектуры. Частотный принцип реализации системных программ основан на выделении в алгоритмах и в обрабатываемых массивах ОС действий и данных по частоте их использования. Следствием применения частотного принципа в современных ОС - наличие многоуровневого планирования при организации работы ОС. Принцип модульности отражает технологические и эксплуатационные свойства системы, предусматривая оформление функционально законченных компонентов ОС в виде отдельных модулей. Принцип функциональной избирательности предусматривает выделение некоторого множества важных модулей, которые должны быть постоянно в “горячем” режиме для обеспечения эффективного управления вычислительным процессом. Этот выделенный набор модулей называют ядром ОС. При формировании состава ядра ОС ищут компромисс между двумя разноречивыми требованиями: в состав ядра должны войти наиболее часто используемые модули; объем

памяти, занимаемый ядром ОС, должен быть как можно меньше. Принцип генерируемости определяет такой способ исходного представления системной программы ОС, который позволяет настраивать эту системную программу исходя из конкретной конфигурации аппаратных средств и круга решаемых проблем. Принцип функциональной избыточности предусматривает обеспечение возможности выполнения одной и той же работы различными средствами. Принцип перемещаемости предусматривает такое построение модулей ОС, при котором результаты работы не зависят от места их расположения. Принцип защиты информации определяет необходимость разработки мер, ограждающих программы и данные пользователя от искажений или нежелательных влияний друг от друга, а также пользователей на ОС и обратно. Принцип независимости программ от внешних устройств заключается в том, что связь программ с конкретными внешними устройствами осуществляется не на уровне подготовки программных устройств (трансляции или компиляции исходного кода, генерации выполняемого модуля), а в период планирования операционной системой ее выполнения. Принцип открытости и наращиваемости ОС предусматривает возможность доступа к ней для анализа пользователями, специалистами, обслуживающим персоналом, а также изменения конфигурации ОС и ее мощности без осуществления процессов генерации.

5. **ОС для автономного компьютера.**

Операционная система компьютера представляет собой комплекс взаимосвязанных программ, который действует как интерфейс между приложениями и пользователями с одной стороны, и аппаратурой компьютера с другой стороны. 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. 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. Таким образом, назначение ОС состоит в предоставлении пользователю некоторой расширенной виртуальной машины, которую легче программировать и с которой легче работать. Управление ресурсами составляет важную часть функций любой операционной системы, в особенности мультипрограммной. В отличие от функций расширенной машины большинство функций управления ресурсами выполняются операционной системой автоматически и прикладному программисту недоступны. 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. Подсистема управления процессами планирует выполнение процессов, занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает синхронизацию процессов, а также обеспечивает взаимодействие между процессами. Функциями ОС по управлению памятью являются отслеживание свободной и занятой памяти; выделение памяти процессам и освобождение памяти при завершении процессов; защита памяти; вытеснение процессов из оперативной памяти на диск и возвращение их в оперативную память, а также настройка адресов программы на конкретную область физической памяти. 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. Поддержание высокоуровневого унифицированного интерфейса прикладного программирования к разнородным устройствам ввода-вывода является одной из наиболее

важных задач ОС. Операционная система должна обеспечивать удобный интерфейс не только для прикладных программ, но и для человека, работающего за терминалом. Прикладному программисту возможности ОС доступны в виде набора функций, составляющих интерфейс прикладного программирования (API).

6. **Функциональные компоненты ОС. Управление процессами.**

Функции операционной системы автономного 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. 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. Таким образом, процесс можно также определить как некоторую заявку на потребление системных ресурсов. 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. Важной задачей операционной системы является 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. Говорят, что каждый процесс работает в своем адресном пространстве, имея в виду защиту адресных пространств, осуществляемую ОС. 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 прогресса. 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, поддерживает синхронизацию процессов, а также обеспечивает взаимодействие между процессами.

7. **Функциональные компоненты ОС. Управление памятью.**

Функции операционной системы автономного 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. Память является для процесса таким же важным ресурсом, как и процессор, так как процесс может выполняться процессором только в том случае, если его коды и данные (не обязательно все) находятся в оперативной памяти. Существует большое разнообразие алгоритмов распределения памяти. Они могут отличаться, например, количеством выделяемых процессу областей памяти (в одних случаях память выделяется процессу в виде одной непрерывной области, а в других - в виде нескольких несмежных областей), степенью свободы границы областей (она может быть жестко зафиксирована на все время существования процесса или же динамически перемещаться при выделении процессу дополнительных объемов памяти). В некоторых системах распределение памяти выполняется страницами фиксированного размера, а в других - сегментами переменной длины. Одним из наиболее популярных способов управления памятью в современных операционных системах является так называемая виртуальная память. Наличие в ОС механизма виртуальной памяти позволяет программисту писать программу так, как будто в его распоряжении имеется однородная оперативная память большого объема, часто существенно превышающего объем имеющейся физической памяти. В действительности все данные, используемые программой, хранятся на диске и при необходимости частями (сегментами или страницами) отображаются в физическую память. Защита памяти - это избирательная способность предохранять выполняемую задачу от записи или чтения памяти, назначенной другой задаче. Средства защиты памяти, реализованные в операционной системе, должны пресекать несанкционированный доступ процессов к чужим областям памяти. Таким образом, функциями ОС по управлению памятью являются отслеживание свободной и занятой памяти; выделение памяти процессам и освобождение памяти при завершении процессов; защита памяти; вытеснение процессов из оперативной памяти на диск, когда размеры основной памяти недостаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место, а также настройка адресов программы на конкретную область физической памяти.

## 8. **Функциональные компоненты ОС. Управление файлами и внешними устройствами.**

Функции операционной системы автономного 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. Способность ОС к «экранированию» сложностей реальной аппаратуры очень ярко проявляется в одной из основных подсистем ОС - файловой системе. Операционная система виртуализирует отдельный набор данных, хранящихся на внешнем накопителе, в виде файла - простой неструктурированной последовательности байтов, имеющей символьное имя. Для удобства работы с данными файлы группируются в каталоги, которые, в свою очередь, образуют группы - каталоги более высокого уровня. Пользователь может с помощью ОС выполнять над файлами и каталогами такие действия, как поиск по имени, удаление, вывод содержимого на внешнее устройство (например, на дисплей), изменение и сохранение содержимого. При выполнении своих функций файловая система тесно взаимодействует с подсистемой управления внешними устройствами, которая по запросам файловой системы осуществляет передачу данных между дисками и оперативной

памятью. Подсистема управления внешними устройствами, называемая также подсистемой ввода-вывода, исполняет роль интерфейса ко всем устройствам, подключенным к компьютеру. Спектр этих устройств очень обширен. Программа, управляющая конкретной моделью внешнего устройства и учитывающая все его особенности, обычно называется драйвером этого устройства.

**9. Функциональные компоненты ОС. Защита данных и администрирование.**

Функции операционной системы автономного 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. 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; ограничивает возможности пользователей в выполнении тех или иных системных действий; может урезать возможности пользовательского интерфейса. Важным средством защиты данных являются функции аудита ОС, заключающиеся в фиксации всех событий, от которых зависит безопасность системы. 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. 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.

**10. Функциональные компоненты ОС. Интерфейс прикладного программиста.**

Функции операционной системы автономного 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. Прикладные программисты используют в своих приложениях обращения кОС, когда для выполнения тех или иных действий им требуется особый статус, которым обладает только операционная система. 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. 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 (ApplicationProgrammingInterface, 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. Приложения выполняют обращения к функциям API с помощью системных вызовов.

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.

**11. Функциональные компоненты ОС. Пользовательский интерфейс.**

Функции операционной системы автономного 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.

Операционная система должна обеспечивать удобный интерфейс не только для прикладных программ, но и для человека, работающего за терминалом. В ранних операционных системах пакетного режима функции пользовательского интерфейса были сведены к минимуму и не требовали наличия терминала. Команды языка управления заданиями набивались на перфокарты, а результаты выводились на печатающее устройство. 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 который 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.

## 12. Требования к современным ОС.

Главным требованием, предъявляемым к операционной системе, является выполнение ею основных функций эффективного управления ресурсами и обеспечение удобного интерфейса для пользователя и прикладных программ. Современная ОС, как правило, должна поддерживать мультипрограммную обработку, виртуальную память, свопинг, многооконный графический интерфейс пользователя, а также выполнять многие другие необходимые функции и услуги. Кроме этих требований функциональной полноты к операционным системам предъявляются не менее важные эксплуатационные требования, которые перечислены ниже. **Расширяемость**. В то время как аппаратная часть компьютера устаревает за несколько лет, полезная жизнь операционных систем может измеряться десятилетиями. Поэтому операционные системы всегда изменяются со временем эволюционно. Изменения ОС обычно заключаются в приобретении ею новых свойств. Если код ОС написан таким образом, что дополнения и изменения могут вноситься без нарушения целостности системы, то такую ОС называют расширяемой. Расширяемость достигается за счет модульной структуры ОС, при которой программы строятся из набора отдельных модулей, взаимодействующих только через функциональный интерфейс. **Переносимость**. В идеале код ОС должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы одного типа на аппаратную платформу другого типа. Переносимые ОС имеют несколько вариантов реализации для разных платформ, такое свойство ОС называют также многоплатформенностъю. **Совместимость**. Для пользователя, переходящего по тем или иным причинам с одной ОС на другую, очень привлекательна возможность запуска в новой операционной системе привычного приложения. Если ОС имеет средства для выполнения прикладных программ, написанных для других операционных систем, то про нее говорят, что она обладает совместимостью с этими ОС. **Надежность и отказоустойчивость**. Система должна быть защищена как от внутренних, так и от внешних ошибок, сбоев и отказов. Ее действия должны быть всегда предсказуемыми, а приложения не должны иметь возможности наносить вред ОС. **Безопасность**. Современная ОС должна защищать данные и другие ресурсы вычислительной системы от несанкционированного доступа. Чтобы ОС обладала свойством безопасности, она должна как минимум иметь в своем составе средства аутентификации - определения легальности пользователей, авторизации - предоставления легальным пользователям дифференцированных прав доступа к ресурсам, аудита - фиксации всех «подозрительных» для безопасности системы событий. **Производительность**. Операционная система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа.

## 13. Ядро и вспомогательные модули ОС.

Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы: ядро - модули, выполняющие основные функции ОС; модули, выполняющие вспомогательные функции ОС.

Модули ядра выполняют такие базовые функции ОС, как управление процессами, памятью, устройствами ввода-вывода и т. п. Ядро составляет сердцевину операционной системы, без него ОС является полностью неработоспособной и не сможет выполнить ни одну из своих функций. В состав ядра входят функции, решающие внутрисистемные задачи организации вычислительного процесса, такие как переключение контекстов, загрузка/выгрузка станиц, обработка прерываний. Эти функции недоступны для приложений. Другой класс функций ядра служит для поддержки приложений, создавая для них так называемую прикладную программную среду. Приложения могут обращаться к ядру с запросами - системными вызовами - для выполнения тех или иных действий, например для открытия и чтения файла, вывода графической информации на дисплей, получения системного времени и т. д. Функции ядра,

которые могут вызываться приложениями, образуют интерфейс прикладного программирования - API.

Функции, выполняемые модулями ядра, являются наиболее часто используемыми функциями операционной системы, поэтому скорость их выполнения определяет производительность всей системы в целом. Для обеспечения высокой скорости работы ОС все модули ядра или большая их часть постоянно находятся в оперативной памяти, то есть являются резидентными.

Остальные модули ОС выполняют весьма полезные, но менее обязательные функции. Такие вспомогательные модули ОС оформляются либо в виде приложений, либо в виде библиотек процедур. Вспомогательные модули ОС обычно подразделяются на следующие группы: 1.утилиты - программы, решающие отдельные задачи управления и сопровождения компьютерной системы, такие, например, как программы сжатия дисков, архивирования данных на магнитную ленту; 2.системные обрабатывающие программы - текстовые или графические редакторы, компиляторы, компоновщики, отладчики; 3.программы предоставления пользователю дополнительных услуг - специальный вариант пользовательского интерфейса, калькулятор и даже игры; 4. библиотеки процедур различного назначения, упрощающие разработку приложений, например библиотека математических функций, функций ввода-вывода и т. д. Как и обычные приложения, для выполнения своих задач утилиты, обрабатывающие программы и библиотеки ОС, обращаются к функциям ядра посредством системных вызовов. Модули ОС, оформленные в виде утилит, системных обрабатывающих программ и библиотек, обычно загружаются в оперативную память только на время выполнения своих функций, то есть являются транзитными. Постоянно в оперативной памяти располагаются только самые необходимые коды ОС, составляющие ее ядро. Такая организация ОС экономит оперативную память компьютера.

**14. Ядро в привилегированном режиме.**

Для надежного управления ходом выполнения приложений операционная система должна иметь по отношению к приложениям определенные привилегии. Иначе некорректно работающее приложение может вмешаться в работу ОС. Операционная система должна обладать исключительными полномочиями также для того, чтобы играть роль арбитра в споре приложений за ресурсы компьютера в мультипрограммном режиме. Ни одно приложение не должно иметь возможности без ведома ОС получать дополнительную область памяти, занимать процессор дольше разрешенного операционной системой периода времени, непосредственно управлять совместно используемыми внешними устройствами. Обеспечить привилегии операционной системе невозможно без специальных средств аппаратной поддержки. Аппаратура компьютера должна поддерживать как минимум два режима работы - пользовательский режим и привилегированный режим, который также называют режимом ядра, или режимом супервизора. Подразумевается, что операционная система или некоторые ее части работают в привилегированном режиме, а приложения - в пользовательском режиме. Наличие в ОС многоуровневой системы привилегий, т. е. более привилегированных и менее привилегированных частей, позволяет повысить устойчивость ОС к внутренним ошибкам программных кодов, так как такие ошибки будут распространяться только внутри модулей с определенным уровнем привилегий. Повышение устойчивости операционной системы, обеспечиваемое переходом ядра в привилегированный режим, достигается за счет некоторого замедления выполнения системных вызовов. Системный вызов привилегированного ядра инициирует переключение процессора из пользовательского режима в привилегированный, а при возврате к приложению - переключение из привилегированного режима в пользовательский. Во всех типах процессоров из-за дополнительной двукратной задержки

переключения переход на процедуру со сменой режима выполняется медленнее, чем вызов процедуры без смены режима.

### 15. Многослойная структура ОС.

Вычислительную систему, работающую под управлением ОС на основе ядра, можно рассматривать как систему, состоящую из трех иерархически расположенных слоев: нижний слой образует аппаратура, промежуточный - ядро, а утилиты, обрабатывающие программы и приложения, составляют верхний слой системы. Слоистую структуру вычислительной системы принято изображать в виде системы концентрических окружностей, иллюстрируя тот факт, что каждый слой может взаимодействовать только со смежными слоями. Поскольку ядро представляет собой сложный многофункциональный комплекс, то многослойный подход обычно распространяется и на структуру ядра. Ядро может состоять из следующих слоев: **Средства аппаратной поддержки ОС**. Часть функций ОС может выполняться аппаратными средствами. Поэтому иногда можно встретить определение операционной системы как совокупности программных и аппаратных средств. К операционной системе относят только средства аппаратной поддержки ОС, то есть те, которые прямо участвуют в организации вычислительных процессов: средства поддержки привилегированного режима, систему прерываний, средства переключения контекстов процессов, средства защиты областей памяти и т. п. **Машинно-зависимые компоненты ОС**. Этот слой образуют программные модули, в которых отражается специфика аппаратной платформы компьютера. В идеале этот слой полностью экранирует вышележащие слои ядра от особенностей аппаратуры. Это позволяет разрабатывать вышележащие слои на основе машинно-независимых модулей, существующих в единственном экземпляре для всех типов аппаратных платформ, поддерживаемых данной ОС. **Базовые механизмы ядра**. Этот слой выполняет наиболее примитивные операции ядра, такие как программное переключение контекстов процессов, диспетчеризацию прерываний, перемещение страниц из памяти на диск и обратно и т. п. Модули данного слоя не принимают решений о распределении ресурсов - они только отрабатывают принятые «наверху» решения, что и дает повод называть их исполнительными механизмами для модулей верхних слоев.**Менеджеры ресурсов**. Этот слой состоит из мощных функциональных модулей, реализующих стратегические задачи по управлению основными ресурсами вычислительной системы. Обычно на данном слое работают менеджеры (называемые также диспетчерами) процессов, ввода-вывода, файловой системы и оперативной памяти. Разбиение на менеджеры может быть и другим. **Интерфейс системных вызовов**. Этот слой является самым верхним слоем ядра и взаимодействует непосредственно с приложениями и системными утилитами, образуя прикладной программный интерфейс операционной системы. Функции API, обслуживающие системные вызовы, предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей их физического расположения.

### 16. Аппаратная зависимость и переносимость ОС. Типовые средства аппаратной поддержки ОС.

Многие операционные системы успешно работают на различных аппаратных платформах без существенных изменений в своем составе. В ОС можно выделить достаточно компактный слой машинно-зависимых компонентов ядра и сделать остальные слои ОС общими для разных аппаратных платформ. Практически все современные аппаратные платформы имеют некоторый типичный набор средств аппаратной поддержки ОС, в который входят следующие компоненты: средства поддержки привилегированного режима; средства трансляции адресов; средства переключения процессов; система прерываний; системный таймер; средства защиты областей памяти.**Средства поддержки привилегированного режима** обычно основаны на системном регистре процессора, часто называемом «словом состояния» машины или процессора. В обязанности средств поддержки привилегированного режима входит выполнение проверки

допустимости выполнения активной программой инструкций процессора при текущем уровне привилегированности. **Средства трансляции адресов** выполняют операции преобразования виртуальных адресов, которые содержатся в кодах процесса, в адреса физической памяти. **Средства переключения процессов** предназначены для быстрого сохранения контекста приостанавливаемого процесса и восстановления контекста процесса, который становится активным. Содержимое контекста обычно включает содержимое всех регистров общего назначения процессора, регистра флагов операций (то есть флагов нуля, переноса, переполнения и т. п.), а также тех системных регистров и указателей, которые связаны с отдельным процессом, а не операционной системой. **Система прерываний** позволяет компьютеру реагировать на внешние события, синхронизировать выполнение процессов и работу устройств ввода-вывода, быстро переходить с одной программы на другую. Механизм прерываний нужен для того, чтобы оповестить процессор о возникновении в вычислительной системе некоторого непредсказуемого события или события, которое не синхронизировано с циклом работы процессора. **Системный таймер**, часто реализуемый в виде быстродействующего регистра-счетчика, необходим операционной системе для выдержки интервалов времени. После того, как время требуемого интервала истекает, таймер инициирует прерывание, которое обрабатывается процедурой операционной системы. Прерывания от системного таймера используются ОС в первую очередь для слежения за тем, как отдельные процессы расходуют время процессора. **Средства защиты областей памяти** обеспечивают на аппаратном уровне проверку возможности программного кода осуществлять с данными определенной области памяти такие операции, как чтение, запись или выполнение (при передачах управления).

### 17. Аппаратная зависимость и переносимость ОС. Машинно-зависимые компоненты ОС.

Многие операционные системы успешно работают на различных аппаратных платформах без существенных изменений в своем составе. В ОС можно выделить достаточно компактный слой машинно-зависимых компонентов ядра и сделать остальные слои ОС общими для разных аппаратных платформ. Одна и та же операционная система не может без каких-либо изменений устанавливаться на компьютерах, отличающихся типом процессора или/и способом организации всей аппаратуры. В модулях ядра ОС не могут не отразиться такие особенности аппаратной платформы, как количество типов прерываний и формат таблицы ссылок на процедуры обработки прерываний, состав регистров общего назначения и системных регистров, состояние которых нужно сохранять в контексте процесса, особенности подключения внешних устройств и многие другие. Однако опыт разработки операционных систем показывает: ядро можно спроектировать таким образом, что только часть модулей будут машинно-зависимыми, а остальные не будут зависеть от особенностей аппаратной платформы. В хорошо структурированном ядре машинно-зависимые модули локализованы и образуют программный слой, естественно примыкающий к слою аппаратуры. Такая локализация машинно-зависимых модулей существенно упрощает перенос операционной системы на другую аппаратную платформу. Объем машинно-зависимых компонентов ОС зависит от того, насколько велики отличия в аппаратных платформах, для которых разрабатывается ОС. Для уменьшения количества машинно-зависимых модулей производители операционных систем обычно ограничивают универсальность машинно-независимых модулей. Это означает, что их независимость носит условный характер и распространяется только на несколько типов процессоров и созданных на основе этих процессоров аппаратных платформ.

### 18. Аппаратная зависимость и переносимость ОС. Переносимость ОС.

Многие операционные системы успешно работают на различных аппаратных платформах без существенных изменений в своем составе. В ОС можно выделить достаточно компактный слой

машинно-зависимых компонентов ядра и сделать остальные слои ОС общими для разных аппаратных платформ. Если код операционной системы может быть сравнительно легко перенесен с процессора одного типа на процессор другого типа и с аппаратной платформы одного типа на аппаратную платформу другого типа, то такую ОС называют переносимой (portable), или мобильной. Хотя ОС часто описываются либо как переносимые, либо как непереносимые, мобильность - это не бинарное состояние, а понятие степени. Вопрос не в том, может ли быть система перенесена, а в том, насколько легко можно это сделать. Для того чтобы обеспечить свойство мобильности ОС, разработчики должны следовать следующим правилам. Большая часть кода должна быть написана на языке, трансляторы которого имеются на всех машинах, куда предполагается переносить систему. Такими языками являются стандартизованные языки высокого уровня. Большинство переносимых ОС написано на языке С, который имеет много особенностей, полезных для разработки кодов операционной системы, и компиляторы которого широко доступны. Объем машинно-зависимых частей кода, которые непосредственно взаимодействуют с аппаратными средствами, должен быть по возможности минимизирован. Так, например, следует всячески избегать прямого манипулирования регистрами и другими аппаратными средствами процессора. Для уменьшения аппаратной зависимости разработчики ОС должны также исключить возможность использования по умолчанию стандартных конфигураций аппаратуры или их характеристик. Аппаратно-зависимый код должен быть надежно изолирован в нескольких модулях, а не быть распределен по всей системе. Изоляции подлежат все части ОС, которые отражают специфику как процессора, так и аппаратной платформы в целом.

**19. Микроядерная архитектура.**

Микроядерная архитектура является альтернативой классическому способу построения операционной системы. Под классической архитектурой в данном случае понимается структурная организация ОС, в соответствии с которой все основные функции операционной системы, составляющие многослойное ядро, выполняются в привилегированном режиме. Суть микроядерной архитектуры состоит в следующем. В привилегированном режиме остается работать только очень небольшая часть ОС, называемая микроядром. Микроядро защищено от остальных частей ОС и приложений. В состав микроядра обычно входят машинно-зависимые модули, а также модули, выполняющие базовые (но не все) функции ядра по управлению процессами, обработке прерываний, управлению виртуальной памятью, пересылке сообщений и управлению устройствами ввода-вывода, связанные с загрузкой или чтением регистров устройств. Все остальные более высокоуровневые функции ядра оформляются в виде приложений, работающих в пользовательском режиме. Менеджеры ресурсов, вынесенные в пользовательский режим, называются серверами ОС, то есть модулями, основным назначением которых является обслуживание запросов локальных приложений и других модулей ОС. Для реализации микроядерной архитектуры необходимым условием является наличие в операционной системе удобного и эффективного способа вызова процедур одного процесса из другого. Поддержка такого механизма и является одной из главных задач микроядра. Работа микроядерной операционной системы соответствует известной модели клиент-сервер, в которой роль транспортных средств выполняет микроядро. Преимущества и недостатки микроядерной архитектуры. Операционные системы, основанные на концепции микроядра, в высокой степени удовлетворяют большинству требований, предъявляемых к современным ОС, обладая переносимостью (весь машинно-зависимый код изолирован в микроядре, поэтому для переноса системы на новый процессор требуется меньше изменений и все они логически сгруппированы вместе), расширяемостью, присущей микроядерной ОС в очень высокой степени, надежностью и создавая хорошие предпосылки для поддержки распределенных приложений. За эти достоинства приходится платить снижением производительности, и это является основным недостатком микроядерной архитектуры. При классической организации

ОС выполнение системного вызова сопровождается двумя переключениями режимов, а при микроядерной организации - четырьмя. Таким образом, операционная система на основе микроядра при прочих равных условиях всегда будет менее производительной, чем ОС с классическим ядром. Именно по этой причине микроядерный подход не получил такого широкого распространения, которое ему предрекали.

### 20. Совместимость и множественные прикладные среды. Двоичная совместимость и совместимость исходных текстов. Трансляция библиотек.

В то время как многие архитектурные особенности операционных систем непосредственно касаются только системных программистов, концепция множественных прикладных сред непосредственно связана с нуждами конечных пользователей - возможностью операционной системы выполнять приложения, написанные для других операционных систем. Такое свойство операционной системы называется совместимостью. Необходимо различать совместимость на двоичном уровне и совместимость на уровне исходных текстов. Двоичная совместимость достигается в том случае, когда можно взять исполняемую программу и запустить ее на выполнение в среде другой ОС. Совместимость на уровне исходных текстов требует наличия соответствующего компилятора в составе программного обеспечения компьютера, на котором предполагается выполнять данное приложение, а также совместимости на уровне библиотек и системных вызовов. При этом необходима перекомпиляция имеющихся исходных текстов в новый исполняемый модуль. Совместимость на уровне исходных текстов важна в основном для разработчиков приложений, в распоряжении которых эти исходные тексты всегда имеются. Но для конечных пользователей практическое значение имеет только двоичная совместимость, так как только в этом случае они могут использовать один и тот же коммерческий продукт, поставляемый в виде двоичного исполняемого кода, в различных операционных средах и на различных машинах. Обладает ли новая ОС двоичной совместимостью или совместимостью исходных текстов с существующими операционными системами, зависит от многих факторов. Самый главный из них - архитектура процессора, на котором работает новая ОС. Если процессор использует тот же набор команд (возможно, с некоторыми добавлениями) и тот же диапазон адресов, тогда двоичная совместимость может быть достигнута довольно просто. Для этого достаточно соблюдения следующих условий: вызовы функций API, которые содержит приложение, должны поддерживаться данной ОС; внутренняя структура исполняемого файла приложения должна соответствовать структуре исполняемых файлов данной ОС. Гораздо сложнее достичь двоичной совместимости операционным системам, предназначенным для выполнения на процессорах, имеющих разные архитектуры. Помимо соблюдения приведенных выше условий необходимо организовать эмуляцию двоичного кода, которая осуществляется с помощью специального программного обеспечения - эмулятора. Эмуляция - это простая, но очень медленная работа. Выходом в таких случаях является использование так называемых прикладных программных сред. Одной из составляющих, формирующих прикладную программную среду, является набор функций интерфейса прикладного программирования API, которые операционная система предоставляет своим приложениям. Для сокращения времени на выполнение чужих программ прикладные среды имитируют обращения к библиотечным функциям. Таким образом достигается существенное ускорение выполнения программ с API другой операционной системы. Иногда такой подход называют трансляцией для того, чтобы отличать его от более медленного процесса эмулирования кода по одной команде за раз. Чтобы программа, написанная для одной ОС, могла быть выполнена в рамках другой ОС, недостаточно лишь обеспечить совместимость API. Концепции, положенные в основу разных ОС, могут входить в противоречие друг с другом. Например, в одной операционной системе приложению может быть разрешено непосредственно управлять устройствами ввода-вывода, в другой - эти действия являются прерогативой ОС. Каждая операционная система имеет свои собственные механизмы защиты ресурсов, свои алгоритмы обработки ошибок и

исключительных ситуаций, особую структуру процесса и схему управления памятью, свою семантику доступа к файлам и графический пользовательский интерфейс. Для обеспечения совместимости необходимо организовать бесконфликтное сосуществование в рамках одной ОС нескольких способов управления ресурсами компьютера.

21. **Совместимость и множественные прикладные среды. Способы реализации программных сред.**

В то время как многие архитектурные особенности операционных систем непосредственно касаются только системных программистов, концепция множественных прикладных сред непосредственно связана с нуждами конечных пользователей - возможностью операционной системы выполнять приложения, написанные для других операционных систем. Такое свойство операционной системы называется совместимостью. Прикладная программная среда - совокупность средств ОС, предназначенная для организации выполнения приложений, использующих определенную систему машинных команд, определенный тип API и определенный формат исполняемой программы. Каждая ОС создает как минимум одну прикладную программную среду. Проблема состоит в обеспечении совместимости нескольких программных сред в рамках одной ОС. При построении множественных прикладных сред используются различные архитектурные решения, концепции эмуляции двоичного кода, трансляции API. Один из наиболее очевидных вариантов реализации множественных прикладных сред основывается на стандартной многоуровневой структуре ОС. Пусть операционная система OS1 поддерживает кроме своих «родных» приложений приложения операционных систем OS2 и OS3. Для этого в ее составе имеются специальные приложения - прикладные программные среды, - которые транслируют интерфейсы «чужих» операционных систем API OS2 и API OS3 в интерфейс своей «родной» операционной системы - API OS1. В другом варианте реализации множественных прикладных сред операционная система имеет несколько равноправных прикладных программных интерфейсов. Пусть операционная система поддерживает приложения, написанные для OS1, OS2 и OS3. Для этого непосредственно в пространстве ядра системы размещены прикладные программные интерфейсы всех этих ОС: API OS1, API OS2 и API OS3. В этом варианте функции уровня API обращаются к функциям нижележащего уровня ОС, которые должны поддерживать все три в общем случае несовместимые прикладные среды.

Еще один способ построения множественных прикладных сред основан на микроядерном подходе. При этом очень важно отделить базовые механизмы операционной системы от специфических для каждой из прикладных сред высокоуровневых функций. В соответствии с микроядерной архитектурой все функции ОС реализуются микроядром и серверами пользовательского режима. Каждая прикладная среда оформляется в виде отдельного сервера пользовательского режима и не включает базовых механизмов. Приложения, используя API, обращаются с системными вызовами к соответствующей прикладной среде через микроядро. Прикладная среда обрабатывает запрос, выполняет его и отсылает приложению результат. Такому подходу к конструированию множественных прикладных сред присущи все достоинства и недостатки микроядерной архитектуры: очень просто можно добавлять и исключать прикладные среды, что является следствием хорошей расширяемости микроядерных ОС; надежность и стабильность выражаются в том, что при отказе одной из прикладных сред все остальные сохраняют работоспособность; низкая производительность микроядерных ОС сказывается на скорости работы прикладных сред, а значит, и на скорости выполнения приложений.

**22. Мультипрограммирование в системах пакетной обработки.**

Мультипрограммирование - способ организации вычислительного процесса, при котором на одном процессоре попеременно выполняются сразу несколько программ. Наиболее

характерными критериями эффективности вычислительных систем являются: пропускная способность; удобство работы пользователей; реактивность. При использовании мультипрограммирования для повышения пропускной способности (количество задач, выполняемых вычислительной системой в единицу времени) компьютера главной целью является минимизация простоев всех устройств компьютера, и, прежде всего центрального процессора. Такая концепция мультипрограммирования положена в основу так называемых пакетных систем. Для достижения этой цели в системах пакетной обработки используется следующая схема функционирования: в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из этого пакета заданий формируется мультипрограммная смесь, то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие разные требования к ресурсам, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины. Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается «выгодное» задание. Следовательно, в вычислительных системах, работающих под управлением пакетных ОС, невозможно гарантировать выполнение того или иного задания в течение определенного периода времени. Рассмотрим совмещение во времени операций ввода-вывода и вычислений. Такое совмещение может достигаться разными способами. Один из них характерен для компьютеров, имеющих специализированный процессор ввода-вывода (канал). Обычно канал имеет систему команд, отличающуюся от системы команд центрального процессора. В системе команд центрального процессора предусматривается специальная инструкция, с помощью которой каналу передаются параметры и указания на то, какую программу ввода-вывода он должен выполнить. Начиная с этого момента центральный процессор и канал могут работать параллельно (а). Другой способ реализуется в компьютерах, в которых внешние устройства управляются не процессором ввода-вывода, а контроллерами. Каждое внешнее устройство имеет свой собственный контроллер, который автономно отрабатывает команды, поступающие от центрального процессора. При этом контроллер и центральный процессор работают асинхронно. Контроллер выполняет свои команды управления устройствами существенно медленнее, чем центральный процессор - свои. Это обстоятельство используется для организации параллельного выполнения вычислений и операций ввода-вывода: в промежутке между передачей команд, контроллеру центральный процессор может выполнять вычисления (б). В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит по инициативе самой активной задачи, поэтому существует высокая вероятность того, что одна задача может надолго занять процессор. Система пакетной обработки повышает эффективность функционирования аппаратуры, но снижает эффективность работы пользователя.

23. **Мультипрограммирование в системах разделения времени и в системах реального времени.**

Повышение удобства и эффективности работы пользователя является целью другого способа мультипрограммирования - разделения времени. В системах разделения времени пользователю предоставляется возможность интерактивной работы сразу с несколькими приложениями. Для этого каждое приложение должно регулярно получать возможность «общения» с пользователем. Это достигается следующим образом: ОС принудительно периодически приостанавливает приложения, не дожидаясь, когда они добровольно освободят процессор. Всем приложениям попеременно выделяется квант процессорного времени, таким образом пользователи, запустившие программы на выполнение, получают возможность поддерживать с ними диалог. Системы разделения времени призваны исправить основной недостаток систем пакетной обработки - изоляцию пользователя-программиста от процесса выполнения его задач. Системы разделения времени обладают меньшей пропускной способностью, чем системы

пакетной обработки. Это вполне соответствует тому, что критерием эффективности систем разделения времени является удобство и эффективность работы пользователя. Еще одна разновидность мультипрограммирования используется в системах реального времени, предназначенных для управления от компьютера различными техническими объектами или технологическими процессами. В этих случаях существует предельно допустимое время, в течение которого должна быть выполнена та или иная управляющая объектом программа. В противном случае может произойти авария. Таким образом, критерием эффективности здесь является способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия). Это время называется временем реакции системы, а соответствующее свойство системы - реактивностью. В системах реального времени мультипрограммная смесь представляет собой фиксированный набор заранее разработанных программ, а выбор программы на выполнение осуществляется по прерываниям или в соответствии с расписанием плановых работ.

24. **Мультипроцессорная обработка.**

Мультипроцессорная обработка - это способ организации вычислительного процесса в системах с несколькими процессорами, при котором несколько задач (процессов, потоков) могут одновременно выполняться на разных процессорах системы. Мультипроцессорные системы часто характеризуют либо как симметричные, либо как несимметричные. При этом следует четко определять, к какому аспекту мультипроцессорной системы относится эта характеристика - к типу архитектуры или к способу организации вычислительного процесса. **Симметричная архитектура** мультипроцессорной системы предполагает однородность всех процессоров и единообразие включения процессоров в общую схему мультипроцессорной системы. Традиционные симметричные мультипроцессорные конфигурации разделяют одну большую память между всеми процессорами. **Масштабируемость** (возможность наращивания числа процессоров) в симметричных системах ограничена вследствие того, что все они пользуются одной оперативной памятью и должны располагаться в одном корпусе. Такая конструкция, называемая масштабируемой по вертикали. **В симметричных архитектурах** обеспечивается достаточно высокая производительность для тех приложений, в которых несколько задач должны активно взаимодействовать между собой. **В асимметричной архитектуре** разные процессоры могут отличаться как своими характеристиками, так и функциональной ролью, которая поручается им в системе. Функциональная неоднородность в асимметричных архитектурах влечет за собой структурные отличия во фрагментах системы, содержащих разные процессоры системы. Масштабирование в асимметричной архитектуре реализуется иначе, чем в симметричной. Так как требование единого корпуса отсутствует, система может состоять из нескольких устройств, каждое из которых содержит один или несколько процессоров. Это масштабирование по горизонтали. Каждое такое устройство называется кластером, а вся мультипроцессорная система - кластерной. **Асимметричное мультипроцессирование** является наиболее простым способом организации вычислительного процесса в системах с несколькими процессорами. Этот способ часто называют также «ведущий-ведомый». Функционирование системы по принципу «ведущий-ведомый» предполагает выделение одного из процессоров в качестве «ведущего», на котором работает операционная система и который управляет всеми остальными «ведомыми» процессорами. Асимметричная организация вычислительного процесса может быть реализована как для симметричной мультипроцессорной архитектуры, так и для несимметричной. В архитектурно-асимметричных системах на роль ведущего процессора может быть назначен наиболее надежный и производительный процессор. **Симметричное мультипроцессирование** как способ организации вычислительного процесса может быть реализовано в системах только с симметричной мультипроцессорной архитектурой. Симметричное мультипроцессирование реализуется общей для всех процессоров операционной системой. При симметричной

организации все процессоры равноправно участвуют и в управлении вычислительным процессом, и в выполнении прикладных задач. В случае отказа одного из процессоров симметричные системы, как правило, сравнительно просто реконфигурируются, что является их большим преимуществом перед плохо реконфигурируемыми асимметричными системами.

**25. Планирование процессов и потоков. Понятия «процесс» и «поток».**

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. Подсистема управления процессами и потоками ответственна за обеспечение процессов необходимыми ресурсами. Синхронизация потоков является одной из важных функций подсистемы управления процессами и потоками. Чтобы поддерживать мультипрограммирование, ОС должна определить и оформить для себя те внутренние единицы работы, между которыми будет 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». В операционных системах, где существуют и процессы, и потоки, процесс рассматривается операционной системой 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. 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.

26. **Создание процессов и потоков.**

Создать процесс - это прежде всего означает создать описатель процесса, в качестве которого выступает одна или несколько информационных структур, содержащих все сведения о процессе, необходимые операционной системе для управления им. Примерами описателей процесса являются блок управления задачей (TCB — TaskControlBlock) в OS/360, управляющий блок процесса (PCB — ProcessControlBlock) в OS/2, дескриптор процесса в UNIX, объект-процесс (object-process) в Windows NT. Создание описателя процесса знаменует собой появление в системе еще одного претендента на вычислительные ресурсы. Начиная с этого момента при распределении ресурсов ОС должна принимать во внимание потребности нового процесса. Создание процесса включает загрузку кодов и данных исполняемой программы данного процесса с диска в оперативную память. При этом подсистема управления процессами тесно взаимодействует с подсистемой управления памятью и файловой системой. При создании потока так же, как и при создании процесса, операционная система генерирует специальную информационную структуру - описатель потока, который содержит идентификатор потока, данные о правах доступа и приоритете, о состоянии потока и другую информацию. Рассмотрим в качестве примера создание процессов в популярной версии операционной системы UNIX System V Release 4. В этой системе потоки не поддерживаются, в качестве единицы управления и единицы потребления ресурсов выступает процесс. При управлении процессами операционная система использует два основных типа информационных структур: дескриптор процесса и контекст процесса. Дескриптор процесса содержит такую информацию о процессе, которая необходима ядру в течение всего жизненного цикла процесса независимо от того,

находится он в активном или пассивном состоянии, находится образ процесса в оперативной памяти или выгружен на диск. Контекст процесса содержит менее оперативную, но более объемную часть информации о процессе, необходимую для возобновления выполнения процесса с прерванного места: содержимое регистров процессора, коды ошибок выполняемых процессором системных вызовов, информация обо всех открытых данным процессом файлах и незавершенных операциях ввода-вывода и другие данные, характеризующие состояние вычислительной среды в момент прерывания. Контекст, так же как и дескриптор процесса, доступен только программам ядра, то есть находится в виртуальном адресном пространстве операционной системы, однако он хранится не в области ядра, а непосредственно примыкает к образу процесса и перемещается вместе с ним, если это необходимо, из оперативной памяти на диск.

**27. Планирование и диспетчеризация потоков.**

На протяжении существования процесса выполнение его потоков может быть многократно прервано и продолжено. Переход от выполнения одного потока к другому осуществляется в результате планирования и диспетчеризации. Работа по определению того, в какой момент необходимо прервать выполнение текущего активного потока и какому потоку предоставить возможность выполняться, называется планированием. Планирование потоков осуществляется на основе информации, хранящейся в описателях процессов и потоков. Планирование потоков, по существу, включает в себя решение двух задач: определение момента времени для смены текущего активного потока; выбор для выполнения потока из очереди готовых потоков. В большинстве операционных систем универсального назначения планирование осуществляется динамически (on-line), то есть решения принимаются во время работы системы на основе анализа текущей ситуации. ОС работает в условиях неопределенности - потоки и процессы появляются в случайные моменты времени и также непредсказуемо завершаются. Динамические планировщики могут гибко приспосабливаться к изменяющейся ситуации и не используют никаких предположений о мультипрограммной смеси. Другой тип планирования — статический — может быть использован в специализированных системах, в которых весь набор одновременно выполняемых задач определен заранее, например в системах реального времени. Планировщик называется статическим (или предварительным планировщиком), если он принимает решения о планировании не во время работы системы, а заранее (off-line). Диспетчеризация заключается в реализации найденного в результате планирования (динамического или статистического) решения, то есть в переключении процессора с одного потока на другой. Прежде чем прервать выполнение потока, ОС запоминает его контекст, с тем чтобы впоследствии использовать эту информацию для последующего возобновления выполнения данного потока. Диспетчеризация сводится к следующему: сохранение контекста текущего потока, который требуется сменить; загрузка контекста нового потока, выбранного в результате планирования; запуск нового потока на выполнение

28. **Состояния потока.**

ОС выполняет планирование потоков, принимая во внимание их состояние. В мультипрограммной системе поток может находиться в одном из трех основных состояний: выполнение - активное состояние потока, во время которого поток обладает всеми необходимыми ресурсами и непосредственно выполняется процессором; ожидание - пассивное состояние потока, находясь в котором, поток заблокирован по своим внутренним причинам (ждет осуществления некоторого события, например завершения операции ввода-вывода, получения сообщения от другого потока или освобождения какого-либо необходимого ему ресурса); готовность - также пассивное состояние потока, но в этом случае поток заблокирован в связи с внешним по отношению к нему обстоятельством (имеет все требуемые для него ресурсы, готов выполняться, однако процессор занят выполнением другого потока). В течение

своей жизни каждый поток переходит из одного состояния в другое в соответствии с алгоритмом планирования потоков, принятым в данной операционной системе.
Рассмотрим типичный граф состояния потока.



Только что созданный поток находится в состоянии готовности, он готов к выполнению и стоит в очереди к процессору. Когда в результате планирования подсистема управления потоками принимает решение об активизации данного потока, он переходит в состояние выполнения и находится в нем до тех пор, пока либо он сам освободит процессор, перейдя в состояние ожидания какого-нибудь события, либо будет принудительно «вытеснен» из процессора, например вследствие исчерпания отведенного данному потоку кванта процессорного времени. В последнем случае поток возвращается в состояние готовности. В это же состояние поток переходит из состояния ожидания, после того как ожидаемое событие произойдет. Рис.2
В состоянии выполнения в однопроцессорной системе может находиться не более одного потока, а в каждом из состояний ожидания и готовности - несколько потоков. Эти потоки образуют очереди соответственно ожидающих и готовых потоков. Очереди потоков организуются путем объединения в списки описателей отдельных потоков. Таким образом, каждый описатель потока, кроме всего прочего, содержит по крайней мере один указатель на другой описатель, соседствующий с ним в очереди. Такая организация очередей позволяет легко их переупорядочивать, включать и исключать потоки, переводить потоки из одного состояния в другое.

**29. Вытесняющие и невытесняющие алгоритмы планирования.**

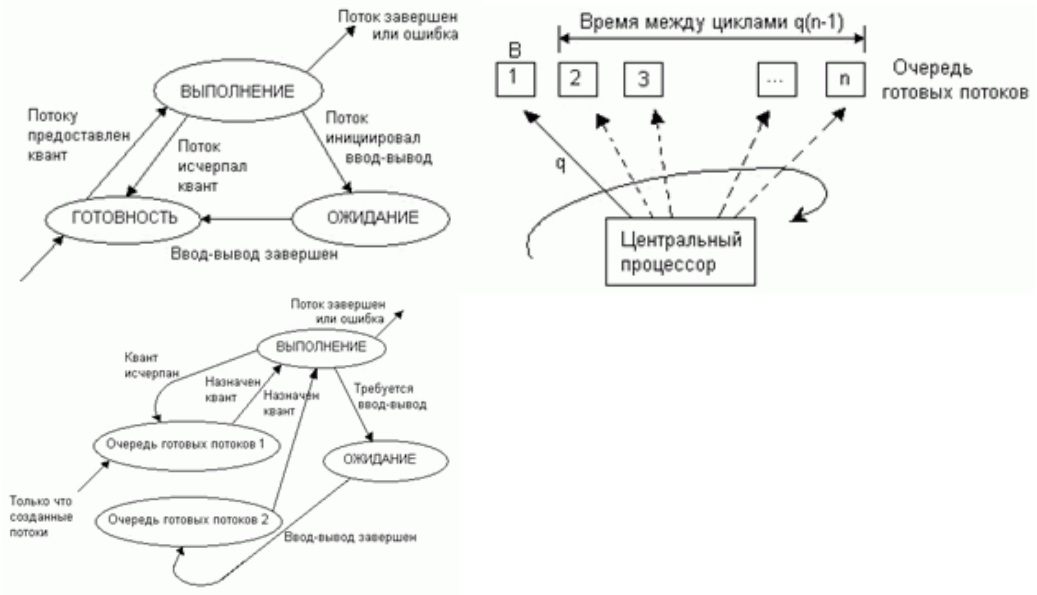
С самых общих позиций все множество алгоритмов планирования можно разделить на два класса: вытесняющие и невытесняющие алгоритмы планирования. Невытесняющие (non-preemptive) алгоритмы основаны на том, что активному потоку позволяется выполняться, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению поток. Вытесняющие (preemptive) алгоритмы - это такие способы планирования потоков, в которых решение о переключении процессора с выполнения одного потока на выполнение другого потока принимается операционной системой, а не активной задачей. Основным различием между вытесняющими и невытесняющими алгоритмами является степень централизации механизма планирования потоков. При применении вытесняющих алгоритмов планирования ОС получает полный контроль над вычислительным процессом, а при применении невытесняющих алгоритмов решения принимаются децентрализованно: активный поток определяет момент смены потоков, а ОС выбирает новый поток для выполнения.

При невытесняющем мультипрограммировании механизм планирования распределен между операционной системой и прикладными программами. Такой механизм создает проблемы как

для пользователей, так и для разработчиков приложений. Для пользователей это означает, что управление системой теряется на произвольный период времени, который определяется приложением (а не пользователем). Поэтому разработчики приложений для операционной среды с невытесняющей многозадачностью вынуждены, возлагая на себя часть функций планировщика, создавать приложения так, чтобы они выполняли свои задачи небольшими частями. Подобный метод работает, но он существенно затрудняет разработку программ и предъявляет повышенные требования к квалификации программиста. Существенным преимуществом невытесняющего планирования является более высокая скорость переключения с потока на поток. Примером эффективного использования невытесняющего планирования являются файл-серверы NetWare 3.x и 4.x, в которых в значительной степени благодаря такому планированию достигнута высокая скорость выполнения файловых операций. Почти во всех современных операционных системах, ориентированных на высокопроизводительное выполнение приложений (UNIX, Windows NT/2000, OS/2, VAX/VMS), реализованы вытесняющие алгоритмы планирования потоков (процессов). В последнее время дошла очередь и до ОС класса настольных систем, например OS/2 Warp и Windows 95/98. При вытесняющем мультипрограммировании функции планирования потоков целиком сосредоточены в операционной системе и программист пишет свое приложение, не заботясь о том, что оно будет выполняться одновременно с другими задачами. При этом операционная система выполняет следующие функции: определяет момент снятия с выполнения активного потока, запоминает его контекст, выбирает из очереди готовых потоков следующий, запускает новый поток на выполнение, загружая его контекст.

30. **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. Рассмотрим случай, когда всем потокам предоставляются кванты одинаковой длины q. Рис.2 Если в системе имеется n потоков, то время, которое поток проводит в ожидании следующего кванта, можно грубо оценить как q(n-l). Чем больше потоков в системе, тем больше время

ожидания, тем меньше возможности вести одновременную интерактивную работу нескольким пользователям. Потоки получают для выполнения квант времени, но некоторые из них используют его не полностью, например из-за необходимости выполнить ввод или вывод данных. В результате возникает ситуация, когда потоки с интенсивными обращениями к вводу-выводу используют только небольшую часть выделенного им процессорного времени. Алгоритм планирования может исправить эту «несправедливость». В качестве компенсации за неиспользованные полностью кванты потоки получают привилегии при последующем обслуживании. Для этого планировщик создает две очереди готовых потоков. Рис.3

31. **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, что приоритет потока тем выше, чем больше (в арифметическом смысле) число, обозначающее приоритет. В других системах, наоборот, чем меньше число, тем выше приоритет. Во многих ОС предусматривается возможность изменения приоритетов в течение жизни потока. Если ОС сама может изменять приоритеты потоков в зависимости от ситуации, складывающейся в системе, то приоритеты называются динамическими в отличие от неизменяемых, фиксированных, приоритетов. Существуют две разновидности приоритетного планирования: обслуживание с относительными приоритетами (а) и обслуживание с абсолютными приоритетами (б).


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 пакетной обработки (в том числе известной ОС OS/360) относительные приоритеты широко используются. А планирование на основе абсолютных приоритетов является подходящим для систем управления объектами, в которых важна быстрая реакция на событие.

**32. 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. Например, в операционной системе UNIX System V Release 4 понятие «поток» отсутствует, и планирование осуществляется на уровне процессов. В этой системе реализована вытесняющая многозадачность, основанная на использовании приоритетов и квантования. А в операционной системе OS/2 планирование основано на использовании квантования и абсолютных динамических приоритетов. Благодаря такому алгоритму планирования в OS/2 ни один поток не будет «забыт» системой и получит достаточно процессорного времени.

33. **Цели и средства синхронизации.**

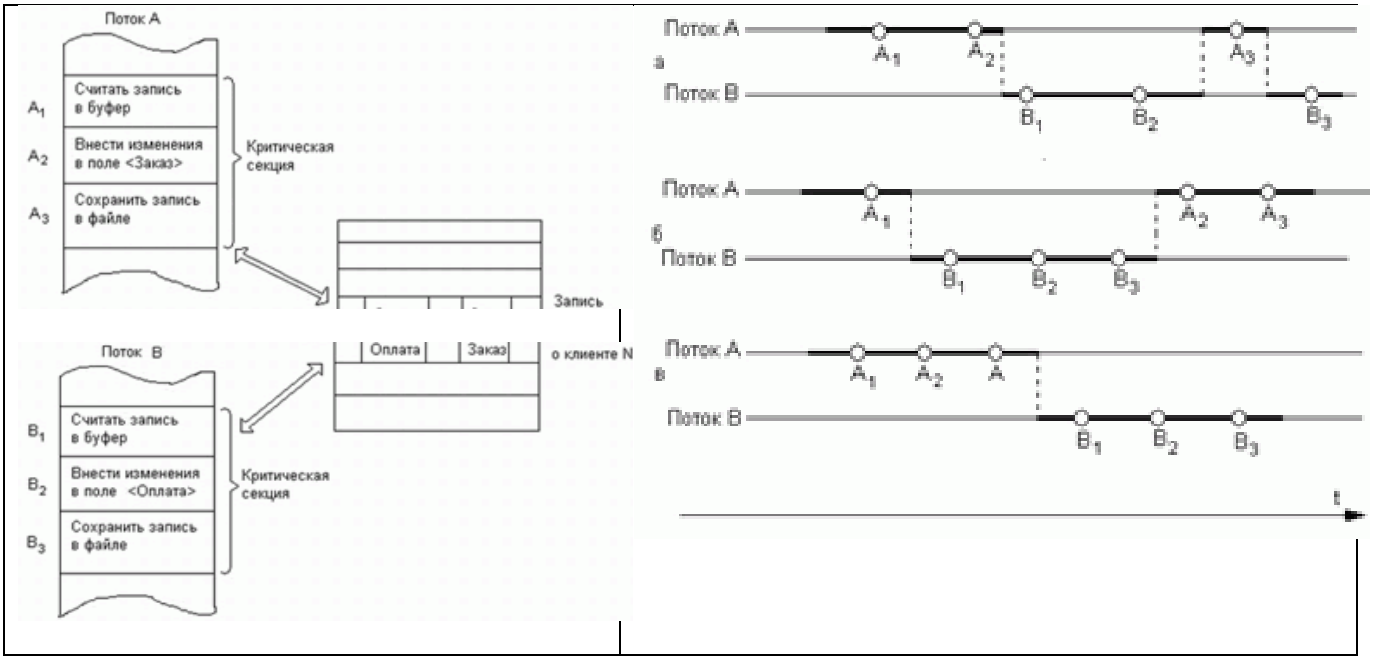
Для синхронизации процессов и потоков, решающих общие задачи и совместно использующих ресурсы, в операционных системах существуют специальные средства: критические секции, семафоры, мьютексы, события, таймеры. Отсутствие синхронизации может приводить к таким нежелательным последствиям, как гонки и тупики. Во многих операционных системах эти средства называются средствами межпроцессного взаимодействия — InterProcessCommunications (IPC). Обычно к средствам IPC относят не только средства межпроцессной синхронизации, но и средства межпроцессного обмена данными. Потребность в синхронизации потоков возникает только в мультипрограммной операционной системе и связана с совместным использованием аппаратных и информационных ресурсов вычислительной системы.

Потоки в общем случае 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. Для синхронизации потоков прикладных программ программист может использовать как собственные средства и приемы синхронизации, так и средства операционной системы. Однако во многих случаях более эффективными или даже единственно возможными являются средства синхронизации, предоставляемые операционной системой в форме системных вызовов. Так, потоки, принадлежащие разным процессам, не имеют возможности вмешиваться каким-либо образом в работу друг друга. Средства синхронизации используются операционной системой не только для синхронизации прикладных процессов, но и для ее внутренних нужд. Обычно разработчики операционных систем предоставляют в распоряжение прикладных и системных программистов широкий спектр средств синхронизации. Эти средства могут образовывать иерархию, когда на основе более простых средств строятся более сложные, а также быть функционально специализированными. Часто функциональные возможности разных системных вызовов синхронизации перекрываются, так что для решения одной задачи программист может воспользоваться несколькими вызовами в зависимости от своих личных предпочтений.

34. **Необходимость синхронизации и гонки.**

Пренебрежение вопросами синхронизации в многопоточной системе может привести к неправильному решению задачи или даже к краху системы. Рассмотрим 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.




Обозначим соответствующие шаги для потока А как Al, А2 и А3, а для потока В как Bl, В2 и В3. Предположим, что в некоторый момент поток А обновляет поле Заказ записи о клиенте N. Для этого он считывает эту запись в свой буфер (шаг А1), модифицирует значение поля Заказ (шаг А2), но внести запись в базу данных (шаг А3) не успевает, так как его выполнение прерывается, например, вследствие завершения кванта времени. Предположим также, что потоку В также потребовалось внести сведения об оплате относительно того же клиента N. Когда подходит очередь потока В, он успевает считать запись в свой буфер (шаг В1) и выполнить обновление поля Оплата (шаг В2), а затем прерывается. Заметим, что в буфере у потока В находится запись о клиенте N, в которой поле Заказ имеет прежнее, не измененное значение.

Когда в очередной раз управление будет передано потоку А, то он, продолжая свою работу, запишет запись о клиенте N с модифицированным полем Заказ в базу данных (шаг А3). После прерывания потока А и активизации потока В последний запишет в базу данных поверх только что обновленной записи о клиенте N свой вариант записи, в которой обновлено значение поля Оплата. Таким образом, в базе данных будут зафиксированы сведения о том, что клиент N произвел оплату, но информация о его заказе окажется потерянной (а). Сложность проблемы синхронизации кроется в нерегулярности возникающих ситуаций. Так, в предыдущем примере можно представить и другое развитие событий: могла быть потеряна информация не о заказе, а об оплате (б) или, напротив, все исправления были успешно внесены (в). Все определяется взаимными скоростями потоков и моментами их прерывания. Поэтому отладка взаимодействующих потоков является сложной задачей. Ситуации, подобные той, когда два или более потоков обрабатывают разделяемые данные и конечный результат зависит от соотношения скоростей потоков, называются гонками.

**35. Критическая секция.**

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. 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.

**36. Блокирующие переменные.**

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. На рисунке показан фрагмент алгоритма потока, использующего для реализации взаимного исключения доступа к критическим данным D блокирующую переменную F(D). 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) снова 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 этого, 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. 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.

**37. Семафоры.**

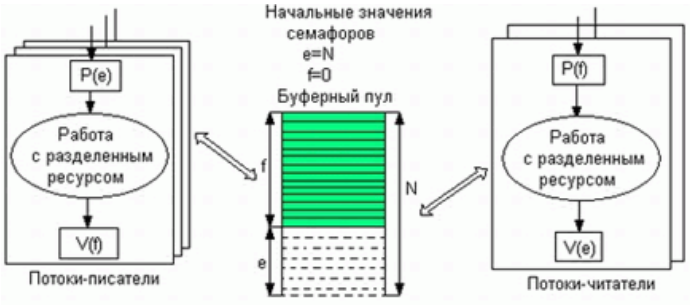
Обобщением блокирующих переменных являются так называемые семафоры Дийкстры. Вместо двоичных переменных Дийкстра предложил использовать переменные, которые могут принимать целые неотрицательные значения. Такие переменные, используемые для синхронизации вычислительных процессов, получили название семафоров.

Для работы с семафорами вводятся два примитива, традиционно обозначаемых P и V. Пусть переменная S представляет собой семафор. Тогда действия V(S) и P(S) определяются следующим образом.

1. V(S): переменная S увеличивается на 1 единым действием. Выборка, наращивание и запоминание не могут быть прерваны. К переменной S нет доступа другим потокам во время выполнения этой операции.
2. P(S): уменьшение S на 1, если это возможно. Если S=0 и невозможно уменьшить S, оставаясь в области целых неотрицательных значений, то в этом случае поток, вызывающий операцию P, ждет, пока это уменьшение станет возможным. Успешная проверка и уменьшение также являются неделимой операцией.

Рассмотрим использование семафоров примере взаимодействия двух выполняющихся в режиме мультипрограммирования потоков, один из которых пишет данные в буферный пул, а другой считывает их из буферного пула. Пусть буферный пул состоит из N буферов, каждый из которых может содержать одну запись. В общем случае поток-писатель и поток-читатель могут иметь различные скорости и обращаться к буферному пулу с переменой интенсивностью. В один период скорость записи может превышать скорость чтения, в другой - наоборот. Для правильной совместной работы поток-писатель должен приостанавливаться, когда все буферы оказываются занятыми, и активизироваться при освобождении хотя бы одного буфера. Напротив, поток-читатель должен приостанавливаться, когда все буферы пусты, и активизироваться при появлении хотя бы одной записи.

Введем два семафора: e - число пустых буферов, и f - число заполненных буферов, причем в исходном состоянии e =N, а f =0. Тогда работа потоков с общим буферным пулом может быть описана следующим образом.




Поток-писатель, прежде всего, выполняет операцию P(e), с помощью которой он проверяет, имеются ли в буферном пуле незаполненные буферы. В соответствии с семантикой операции P, если семафор e равен 0, то поток-писатель переходит в состояние ожидания. Если же значением e является положительное число, то он уменьшает число свободных буферов, записывает

данные в очередной свободный буфер и после этого наращивает число занятых буферов операцией V(f). Поток-читатель действует аналогичным образом, с той разницей, что он начинает работу с проверки наличия заполненных буферов, а после чтения данных наращивает количество свободных буферов. В данном случае предпочтительнее использовать семафоры вместо блокирующих переменных. Действительно, критическим ресурсом здесь является буферный пул, который может быть представлен как набор идентичных ресурсов - отдельных буферов, а значит, с буферным пулом могут работать сразу несколько потоков, и именно столько, сколько буферов в нем содержится. Использование двоичной переменной не позволяет организовать доступ к критическому ресурсу более чем одному потоку. Семафор же решает задачу синхронизации более гибко, допуская к разделяемому пулу ресурсов заданное количество потоков. Так, в нашем примере с буферным пулом могут работать максимум N потоков, часть из которых может быть «писателями», а часть - «читателями».

**38. Тупики.**

Одной из проблем синхронизации являются взаимные блокировки, называемые также дедлоками, клинчами, или тупиками.


Рассмотрим пример тупика. Пусть двум потокам, принадлежащим разным процессам и выполняющимся в режиме мультипрограммирования, для выполнения их работы нужно два ресурса, например принтер и последовательный порт.

На рисунке показаны фрагменты соответствующих программ. Поток А запрашивает сначала принтер; а затем порт, а поток В запрашивает устройства в обратном порядке. Предположим, что после того, как ОС назначила принтер потоку А и установила связанную с этим ресурсом блокирующую переменную, поток А был прерван. Управление получил поток В, который сначала выполнил запрос на получение COM-порта, затем при выполнении следующей команды был заблокирован, так как принтер оказался уже занятым потоком А. Управление снова получил поток А, который в соответствии со своей программой сделал попытку занять порт и был заблокирован, поскольку порт уже выделен потоку В. В таком положении потоки А и В могут находиться сколь угодно долго.

В зависимости от соотношения скоростей потоков они могут либо взаимно блокировать друг друга, либо образовывать очереди к разделяемым ресурсам, либо совершенно независимо использовать разделяемые ресурсы.

В примере тупик был образован двумя потоками, но взаимно блокировать друг друга может и большее число потоков.

Невозможность потоков завершить начатую работу из-за возникновения взаимных блокировок снижает производительность вычислительной системы. Поэтому проблеме предотвращения тупиков уделяется большое внимание.

Тупики могут быть предотвращены на стадии написания программ, то есть программы должны быть написаны таким образом, чтобы тупик не мог возникнуть при любом соотношении взаимных скоростей потоков.

В тех же случаях, когда тупиковую ситуацию не удалось предотвратить, важно быстро и точно ее распознать, поскольку блокированные потоки не выполняют никакой полезной работы. Существуют формальные, программнореализованные методы распознавания тупиков,

основанные на ведении таблиц распределения ресурсов и таблиц запросов к занятым ресурсам. Анализ этих таблиц позволяет обнаружить взаимные блокировки.

39. **Синхронизирующие объекты ОС.**

Примерами синхронизирующих объектов ОС являются системные семафоры, мьютексы, события, таймеры и другие - их набор зависит от конкретной ОС, которая создает эти объекты по запросам процессов. Кроме того, для синхронизации могут быть использованы такие «обычные» объекты ОС, как файлы, процессы и потоки. Все эти объекты могут находиться в двух состояниях: сигнальном и несигнальном — свободном. Для каждого объекта смысл, вкладываемый в понятие «сигнальное состояние», зависит от типа объекта. Поток переходит в сигнальное состояние тогда, когда он завершается. Процесс переходит в сигнальное состояние тогда, когда завершаются все его потоки. Файл переходит в сигнальное состояние в том случае, когда завершается операция ввода-вывода для этого файла. Для остальных объектов сигнальное состояние устанавливается в результате выполнения специальных системных вызовов. Потоки с помощью специального системного вызова сообщают операционной системе о том, что они хотят синхронизировать свое выполнение с состоянием некоторого объекта. Будем далее называть этот системный вызов Wait(X), где X - указатель на объект синхронизации. Системный вызов, с помощью которого поток может перевести объект синхронизации в сигнальное состояние, назовем Set(X). Поток, выполнивший системный вызов Wait(X), переводится операционной системой в состояние ожидания до тех пор, пока объект X не перейдет в сигнальное состояние. Примерами системных вызовов типа Wait() и Set() являются вызовы WaitForSingleObject() и SetEvent() в Windows NT, DosSenWait() и OosSemSet() в OS/2, sleep() и wakeup() в UNIX. В ОС имеются и другие, более универсальные объекты синхронизации, такие как событие (event), мъютекс (nmtex), системный семафор и другие. Мьютекс, как и семафор, обычно используется для управления доступом к данным. В отличие от объектов-потоков, объектов-процессов и объектов-файлов, которые при переходе в сигнальное состояние переводят в состояние готовности все потоки, ожидающие этого события, объект - мьютекс «освобождает» из очереди ожидающих только один поток.

Пусть поток, который, пытаясь получить доступ к критическим данным, выполнил системный вызов Wait(X), где X - указатель на мьютекс. Пусть мьютекс находится в сигнальном состоянии, в этом случае поток тут же становится его владельцем, устанавливая его в несигнальное состояние, и входит в критическую секцию. После того как поток выполнил работу с критическими данными, он «отдает» мьютекс, устанавливая его в сигнальное состояние. В этот момент мьютекс свободен и не принадлежит ни одному потоку. Если какой-либо поток ожидает его освобождения, то он становится следующим владельцем этого мьютекса, одновременно мьютекс переходит в несигнальное состояние. Объект-событие обычно используется не для доступа к данным, а для того, чтобы оповестить другие потоки о том, что некоторые действия завершены.

**40. Сигналы.**

Сигнал дает возможность задаче реагировать на событие, источником которого может быть операционная система или другая задача. Сигналы вызывают прерывание задачи и выполнение заранее предусмотренных действий. Сигналы могут вырабатываться синхронно, то есть как результат работы самого процесса, а могут быть направлены процессу другим процессом; то есть вырабатываться асинхронно. В системе может быть определен набор сигналов. Программный код процесса, которому поступил сигнал, может либо проигнорировать его, либо прореагировать на него стандартным действием, либо выполнить специфические действия, определенные прикладным программистом. В последнем случае в программном коде необходимо предусмотреть специальные системные вызовы, с помощью которых операционная

система информируется, какую процедуру надо выполнить в ответ на поступление того или иного сигнала. Сигналы обеспечивают логическую связь между процессами, а также между процессами и пользователями. Поскольку посылка сигнала предусматривает знание идентификатора процесса, то взаимодействие посредством сигналов возможно только между родственными процессами, которые могут получить данные об идентификаторах друг друга.

## 3. Методические материалы, определяющие процедуру и критерии оценивания сформированности компетенций при проведении промежуточной аттестации

### Критерии формирования оценок по ответам на вопросы, выполнению тестовых заданий

- оценка **«отлично»** выставляется обучающемуся, если количество правильных ответов на вопросы составляет 100 – 90% от общего объёма заданных вопросов;

- оценка **«хорошо»** выставляется обучающемуся, если количество правильных ответов на вопросы – 89 – 76% от общего объёма заданных вопросов;

- оценка **«удовлетворительно»** выставляется обучающемуся, если количество правильных ответов на тестовые вопросы –75–60 % от общего объёма заданных вопросов;

- оценка **«неудовлетворительно»** выставляется обучающемуся, если количество правильных ответов – менее 60% от общего объёма заданных вопросов.

### Критерии формирования оценок по результатам выполнения заданий

**«Отлично/зачтено»** – ставится за работу, выполненную полностью без ошибок и недочетов.

**«Хорошо/зачтено»** – ставится за работу, выполненную полностью, но при наличии в ней не более одной негрубой ошибки и одного недочета, не более трех недочетов.

**«Удовлетворительно/зачтено»** – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов.

**«Неудовлетворительно/не зачтено»** – ставится за работу, если число ошибок и недочетов превысило норму для оценки «удовлетворительно» или правильно выполнено менее 2/3 всей работы.

*Виды ошибок:*

- *грубые ошибки: незнание основных понятий, правил, норм; незнание приемов решения задач; ошибки, показывающие неправильное понимание условия предложенного задания.*

- *негрубые ошибки: неточности формулировок, определений; нерациональный выбор хода решения.*

- *недочеты: нерациональные приемы выполнения задания; отдельные погрешности в формулировке выводов; небрежное выполнение задания.*

## Критерии формирования оценок по зачету с оценкой

**«Отлично/зачтено» –** студент приобрел необходимые умения и навыки, продемонстрировал навык практического применения полученных знаний, не допустил логических и фактических ошибок

**«Хорошо/зачтено»** – студент приобрел необходимые умения и навыки, продемонстрировал навык практического применения полученных знаний; допустил незначительные ошибки и неточности.

**«Удовлетворительно/зачтено»** – студент допустил существенные ошибки.

**«Неудовлетворительно/не зачтено»** – студент демонстрирует фрагментарные знания изучаемого курса; отсутствуют необходимые умения и навыки, допущены грубые ошибки.

## Критерии формирования оценок по экзамену

«**Отлично**» (5 баллов) – обучающийся демонстрирует знание всех разделов изучаемой дисциплины: содержание базовых понятий и фундаментальных проблем; умение излагать программный материал с демонстрацией конкретных примеров. Свободное владение материалом должно характеризоваться логической ясностью и четким видением путей применения полученных знаний в практической деятельности, умением связать материал с другими отраслями знания.

«**Хорошо**» (4 балла) – обучающийся демонстрирует знания всех разделов изучаемой дисциплины: содержание базовых понятий и фундаментальных проблем; приобрел необходимые умения и навыки, освоил вопросы практического применения полученных знаний, не допустил фактических ошибок при ответе, достаточно последовательно и логично излагает теоретический материал, допуская лишь незначительные нарушения последовательности изложения и некоторые неточности. Таким образом данная оценка выставляется за правильный, но недостаточно полный ответ.

«**Удовлетворительно**» (3 балла) – обучающийся демонстрирует знание основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. Однако знание основных проблем курса не подкрепляются конкретными практическими примерами, не полностью раскрыта сущность вопросов, ответ недостаточно логичен и не всегда последователен, допущены ошибки и неточности.

«**Неудовлетворительно**» (0 баллов) – выставляется в том случае, когда обучающийся демонстрирует фрагментарные знания основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. У экзаменуемого слабо выражена способность к самостоятельному аналитическому мышлению, имеются затруднения в изложении материала, отсутствуют необходимые умения и навыки, допущены грубые ошибки и незнание терминологии, отказ отвечать на дополнительные вопросы, знание которых необходимо для получения положительной оценки.

Экспертный лист
оценочных материалов для проведения промежуточной аттестации по
дисциплине «Операционные сисемы»

по направлению подготовки/специальности

**09.03.01 «Информатика и вычислительная техника»**
*(код и наименование)*

Направленность (профиль)/специализация

*(наименование)*

Бакалавар
квалификация выпускника

| 1. Формальное оценивание | | | |
|---|---|---|---|
| Показатели | | Присутствуют | Отсутствуют |
| Наличие обязательных структурных элементов: | | + | |
| –титульный лист | | + | |
| –пояснительная записка | | + | |
| – типовые оценочные материалы | | + | |
| –методические материалы, определяющие процедуру и критерии оценивания | | + | |
| Содержательное оценивание | | | |
| Показатели | Соответствует | Соответствует частично | Не соответствует |
| Соответствие требованиям ФГОС ВОк результатам освоения программы | + | | |
| Соответствие требованиям ОПОП ВОк результатам освоения программы | + | | |
| Ориентация на требования к трудовым функциям ПС (при наличии утвержденного ПС) | + | | |
| Соответствует формируемым компетенциям, индикаторам достижения компетенций | + | | |

Заключение: ФОС рекомендуется/ не рекомендуется к внедрению; обеспечивает/ не обеспечивает объективность и достоверность результатов при проведении оценивания результатов обучения; критерии и показатели оценивания компетенций, шкалы оценивания обеспечивают/ не обеспечивают проведение всесторонней оценки результатов обучения.