РОСЖЕЛДОР
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)
Тихорецкий техникум железнодорожного транспорта – филиал РГУПС
(ТТЖТ – филиал РГУПС)

А.В. Украинский

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по выполнению практических занятий по дисциплине

Операционные системы и среды

для студентов специальности

09.02.01 Компьютерные системы и комплексы

III курса

Тихорецк, 2024

УТВЕРЖДАЮ
Заместитель директора по УР
Н.Ю. Шитикова

Методические указания по выполнению практических занятий по дисциплине «Операционные системы и среды» разработаны в соответствии с рабочей учебной программой для специальности 09.02.01 Компьютерные системы и комплексы.

Организация-разработчик: Тихорецкий техникум железнодорожного транспорта – филиал Федерального государственного бюджетного образовательного учреждения высшего образования «Ростовский государственный университет путей сообщения» (ТТЖТ – филиал РГУПС).

Составитель:
Украинский А.В., преподаватель ТТЖТ – филиала РГУПС

Рекомендованы цикловой комиссией № 4 специальностей 09.02.01, 11.02.06, 38.02.01.

Протокол заседания № 1 от «02» сентября 2024 г.

Содержание

Пояснительная записка

Настоящие методические рекомендации устанавливают требования к выполнению практических занятий.

Методические рекомендации к практическим занятиям составлены в соответствии с рабочей учебной программой по дисциплине **«Операционные системы и среды»** и предназначены для студентов 3 курса специальности 09.02.01 Компьютерные системы и комплексы.

Курс содержит практические занятия, соответствующие лекционному материалу. Основной целью курса является изучение операционных систем разных семейств.

Каждое практическое занятие по курсу содержит название, цели работы, теоретический материал, задания и контрольные вопросы. В методических рекомендациях подробно описан ход выполнения работы.

Курс практических занятий рассчитан на 30 часов.

Практические занятия выполняются студентами индивидуально в отдельных тетрадях.

В результате выполнения работ студенты закрепляют знания, полученные на теоретических занятиях и отрабатывают навыки эффективного применения современных операционных систем в различных сферах деятельности.

Рекомендации предназначены для повышения качества и облегчения процесса выполнения заданий.

Практическое занятие № 1

Тема: Работа в оболочке командной строки. CMD

Цель занятия:

Знакомство с основами работы в командной строке CMD, научиться выполнять основные команды и использовать их для управления файлами и системными настройками.

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

Командная строка Windows (**CMD**) — это текстовая интерфейсная оболочка, которая позволяет пользователям вводить команды для выполнения различных операций на компьютере.

1. Запуск командной строки:
   - Способы запуска: через меню "Пуск", с помощью комбинации Win + R (ввести "**cmd**"), или через поиск.
2. Основные команды:
   - `dir` — отображает список файлов и папок в текущем каталоге.
   - `cd` — изменяет текущий каталог.
   - `mkdir` — создает новую папку.
   - `rmdir` — удаляет пустую папку.
   - `del` — удаляет файлы.
   - `copy` — копирует файлы.
   - `move` — перемещает файлы.
   - `ipconfig` — отображает настройки сети.
   - `ping` — проверяет соединение с другим компьютером или адресом.
3. Работа с файлами и папками:
   - Использование относительных и абсолютных путей.
   - Время и дата: команды `time` и `date`.
4. Использование параметров команд:
   Большинство команд поддерживают параметры, например, `dir /w` для отображения списка в широком виде.
   Дополнительные команды CMD:
   - **type**: Просмотр содержимого текстового файла.
   - **echo**: Вывод текста на экран.
   - **cls**: Очистка экрана.
   - **help**: Получение справки по командам.

✍Задания для студентов

1. **Навигация по файловой системе**:
   - Откройте командную строку.
   - Перейдите в директорию `C:\Users\<Ваше имя>\Documents`.
   - Создайте новую директорию с именем `Test`.
   - Перейдите в созданную директорию.
   - Создайте текстовый файл с именем `example.txt`.

- Откройте файл `example.txt` и добавьте в него текст "Пример текста".

2. **Работа с файлами**:
   - Скопируйте файл **example.txt** в директорию **C:\Users\<Ваше имя>\Documents\Test\Backup**.
   - Переместите файл **example.txt** в директорию **C:\Users\<Ваше имя>\Documents\Test\Archive**.
   - Удалите файл **example.txt** из директории **C:\Users\<Ваше имя>\Documents\Test\Archive**.

3. **Создание пакетного файла**:
   - Создайте пакетный файл с именем **script.bat**.
   - Добавьте в файл следующие команды:

```
@echo off
echo Создание директории...
mkdir C:\Users\<Ваше имя>\Documents\Test\NewFolder
echo Директория создана.
pause
```

   - Выполните пакетный файл и проверьте, что директория была создана.

Дополнительные Задания:

1. Навигация по файловой системе:
   - Откройте CMD и выполните команды для создания новой папки и перехода в неё.
2. Работа с файлами:
   - Создайте текстовый файл (например, `student.txt`), скопируйте его в другую папку, а затем удалите оригинал.
3. Отображение сетевых настроек:
   Выполните команду `ipconfig` и сделайте снимок экрана с результатами.
4. Использование команды **ping**:
   - Пинговать популярные веб-сайты (например, yandex.ru) и записать результаты.
5. Создание сценария:
   - Напишите простой BAT-файл, который создает две папки и копирует в них файлы.

Примеры команд

1. Создание новой папки:

```
mkdir MyFolderStudent
```

2. Копирование файла:

```
copy D:\Source\FIO.txt D:\MyFolderStudent\
```

3. Удаление файла:

```
del D:\Folder\File.txt
```

4. Пингование адреса:

```
ping yandex.ru
```

Дополнительные примеры:

1. **Пример команды для просмотра содержимого директории**:
   dir C:\Users\<Ваше имя>\Documents
2. **Пример команды для копирования файла**:
   copy C:\Users\<Ваше имя>\Documents\example.txt C:\Users\<Ваше имя>\Documents\Test\Backup
3. **Пример команды для перемещения файла**:
   move C:\Users\<Ваше имя>\Documents\example.txt C:\Users\<Ваше имя>\Documents\Test\Archive

В результате занятия был получен практический навык работы с командной строкой Windows, что поможет более эффективно управлять системой и выполнять различные задачи через текстовый интерфейс.

Содержание отчета

1. Наименование, цель занятия, дата;
2. Выполнение заданий;
3. Ответы на контрольные вопросы;
4. Вывод о проделанной работе.

🕮 Контрольные вопросы:

1. Как запустить командную строку в Windows?
2. Что делает команда `cd` и как её использовать для навигации?
3. Каковы основные отличия между командами `copy` и `move`?
4. Как вывести список файлов в папке с помощью CMD?
5. Как изучить доступные параметры для команды, например, `dir`?
6. Что такое командная строка и зачем она нужна?
7. Какие основные команды используются для навигации по файловой системе?
8. Как создать новую директорию в командной строке?
9. Как скопировать файл из одной директории в другую?
10. Как создать и выполнить пакетный файл?
11. Как просмотреть содержимое текстового файла в командной строке?
12. Как удалить файл или директорию в командной строке?

Практическое занятие № 2

Тема: Работа в оболочке командной строки. PowerShell

Цель занятия:

Научиться основам работы в PowerShell, включая выполнение команд, скриптов и управление системными настройками.

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

PowerShell — это мощная оболочка командной строки и язык сценариев от Microsoft, предназначенный для автоматизации задач и управления системой. В отличие от традиционной командной строки, PowerShell работает с объектами, а не с текстом.

1. Запуск **PowerShell**:
   - Запуск через меню "Пуск" (ввести "PowerShell").
   - Использование сочетания клавиш Win + X и выбор пункта "Windows PowerShell" или "Windows Terminal".

2. Основные команды (**cmdlets**):
   - `Get-Command` — отображает доступные команды.
   - `Get-Help` — получение информации о командах.
   - `Get-Process` — отображает список запущенных процессов.
   - `Get-Service` — отображает список сервисов.
   - `Set-ExecutionPolicy` — управление политикой выполнения скриптов.
   - `Copy-Item` — копирование файлов и папок.
   - `Remove-Item` — удаление файлов и папок.
   - `New-Item` — создание новых файлов или папок.

3. Дополнительные команды **PowerShell**:
   - `Get-ChildItem`: Просмотр содержимого директории.
   - `Set-Location`: Смена текущей директории.
   - `Move-Item`: Перемещение файлов.
   - `Get-Content`: Просмотр содержимого текстового файла.
   - `Write-Output`: Вывод текста на экран.
   - `Clear-Host`: Очистка экрана.

4. Работа с объектами:
   - PowerShell возвращает данные в виде объектов, что позволяет использовать фильтрацию и преобразование данных. Например, `Get-Process | Where-Object {$_.CPU -gt 1}` отобразит процессы с использованием CPU более 1%.

5. Создание и выполнение скриптов:
   - Скрипты PowerShell имеют расширение `.ps1`. Их можно запускать из PowerShell, предварительно установив соответствующую политику выполнения.

✍Задания для студентов

1. **Навигация по файловой системе**:
   - Откройте PowerShell.
   - Перейдите в директорию **D:\PowerShell\<Название группы>\Doc**.
   - Создайте новую директорию с именем **Test**.
   - Перейдите в созданную директорию.
   - Создайте текстовый файл с именем **example.txt**.
   - Откройте файл **example.txt** и добавьте в него текст "Пример текста".

2. **Работа с файлами**:
   - Скопируйте файл **example.txt** в директорию **D:\PowerShell\<Название группы>\Doc\Test\Backup**.
   - Переместите файл **example.txt** в директорию **D:\PowerShell\<Название группы>\Doc\Test\Archive**.
   - Удалите файл **example.txt** из директории **D:\PowerShell\<Название группы>\Doc\Test\Archive**.

3. **Создание скрипта PowerShell**:
   - Создайте скрипт PowerShell с именем **script.ps1**.
   - Добавьте в файл следующие команды:

```
Write-Output "Создание директории..."
New-Item -Path " D:\PowerShell\<Название группы>\Doc\Test\NewFolder" -ItemType
Directory
Write-Output "Директория создана."
```

   - Выполните скрипт и проверьте, что директория была создана.

Дополнительные задания:

1. Получение справки:
   - Используйте команду `Get-Help` для получения информации по команде `Get-Process.`
2. Работа с процессами:
   - Выведите список всех запущенных процессов и сохраните его в текстовый файл.
3. Создание файловой структуры:
   - Создайте папку "**Test**" и внутри неё — три текстовых файла с произвольным содержимым.
4. Управление службами:
   - Получите список всех служб и найдите службу, содержащую “Windows”. Запишите её статус.
5. Написание скрипта:
   - Создайте скрипт, который будет очищать временные файлы в указанной папке и выводить количество удалённых файлов.

🎼 Примеры команд

1. **Пример команды для просмотра содержимого директории**:

```
Get-ChildItem -Path "D:\PowerShell\<Название группы>\Doc"
```

2. **Пример команды для копирования файла**:

```
Copy-Item -Path " D:\PowerShell\<Название группы>\Doc\example.txt" -Destination
"D:\PowerShell\<Название группы>\Doc\Test\Backup"
```

3. **Пример команды для перемещения файла**:

```
Move-Item -Path "D:\PowerShell\<Название группы>\Doc\example.txt" -Destination
"D:\PowerShell\<Название группы>\Doc\Test\Archive"
```

Дополнительные примеры:

1. Получение списка процессов:

```
Get-Process
```

2. Копирование файла:

```
Copy-Item C:\Source\File.txt C:\Destination\
```

3. Удаление файла:

```
Remove-Item C:\Folder\File.txt
```

4. Получение информации о службах:

```
Get-Service
```

5. Создание нового текстового файла:

```
New-Item -Path C:\Test\ -Name "MyFile.txt" -ItemType "file"
```

После завершения занятия студенты смогут уверенно работать с PowerShell, что позволит им эффективно автоматизировать задачи и управлять системными ресурсами, используя мощный инструмент командной строки.Этот материал также поможет студентам освоить основные команды и возможности PowerShell, научит их автоматизировать задачи с помощью скриптов.

Содержание отчета

1. Наименование, цель занятия, дата;
2. Выполнение заданий;
3. Ответы на контрольные вопросы;
4. Вывод о проделанной работе.

🕮 Контрольные вопросы:

1. Каковы основные отличия между командной строкой и PowerShell?
2. Как можно получить справку о команде в PowerShell?
3. Что такое cmdlet и приведите примеры?
4. Как вывести процессы с использованием CPU более 2%?
5. Какова функция команды Set-ExecutionPolicy?
6. Что такое PowerShell и зачем она нужна?
7. Какие основные команды используются для навигации по файловой системе?
8. Как создать новую директорию в PowerShell?
9. Как скопировать файл из одной директории в другую?
10. Как создать и выполнить скрипт PowerShell?
11. Как просмотреть содержимое текстового файла в PowerShell?
12. Как удалить файл или директорию в PowerShell?

Практическое занятие № 3

Тема: Установка и предварительная настройка ОС

Цель занятия:

Научиться процессу установки операционной системы (ОС), предварительной настройкой и базовыми операциями после установки.

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

1. Подготовка к установке ОС:
    - Системные требования: ознакомление с минимальными и рекомендуемыми системными требованиями для устанавливаемой ОС.
    - Подбор установки: выбор версии ОС (например, Windows, Linux) в зависимости от целей использования (домашний компьютер, сервер, разработка и т. д.).
    - Создание загрузочного носителя: использование инструментов (например, Rufus, UNetbootin) для создания загрузочной флешки или DVD-диска.
2. Процесс установки ОС:
    - Загрузка с носителя: изменение порядка загрузки в BIOS/UEFI для загрузки с USB или DVD (при наличии привода).
    - Этапы установки:
        - Выбор языка и региона
        - Принятие лицензионного соглашения
        - Разделение диска (если требуется)
        - Выбор места установки
        - Завершение установки и перезагрузка.
3. Первоначальная настройка после установки:
    - Создание учетной записи пользователя (локальной или Microsoft для Windows).
    - Установка обновлений: важность своевременного обновления операционной системы.
    - Установка необходимых программ: антивирус, браузер, офисные приложения и др.
    - Настройка параметров безопасности: настройки брандмауэра и антивируса.
4. Знакомство с интерфейсом ОС:
    - Обзор рабочего стола, меню "Пуск", панели задач.
    - Изучение базовых функций: создание и удаление файлов и папок, использование панели управления.

✍Задания для студентов

1. Создание загрузочного носителя:
    - С помощью инструмента Rufus создайте загрузочную флешку с образом операционной системы (например, Windows или Ubuntu).
2. Установка ОС:
    - Проведите установку ОС на виртуальной машине (например, с помощью VirtualBox или VMware) с созданным загрузочным носителем.

3. Первоначальная настройка:
    - Настройте учетную запись пользователя и выполните поиск обновлений. Запишите шаги и результаты.
4. Установка программного обеспечения:
    - Установите не менее трёх полезных приложений (например, браузер, текстовый редактор, антивирус) и выполните их настройку.
    - Установите драйверы для всех устройств, перейдя на сайт производителя оборудования и скачав необходимые драйверы
    - Настройте сетевые параметры (подключение к Wi-Fi или проводной сети).
5. Создание резервной копии:
    - Настройте систему резервного копирования (например, с помощью встроенных инструментов ОС) и выполните первую резервную копию системы.

🎵 Примеры команд

1. Создание загрузочного носителя:
    - Выберите ISO-образ Windows или Linux в Rufus и настройте параметры: файловая система, метка тома, загрузочный носитель.
2. Установка обновлений (Windows):

```
Настройки -> Обновление и безопасность -> Проверка наличия обновлений
```

3. Установка программ:
    - Для Windows: скачать установочный файл с официального сайта.
    - Для Ubuntu/Linux: использование терминала:

```
sudo apt update
sudo apt install firefox
```

В результате занятия студенты получат практические навыки по установке и первоначальной настройке операционной системы, что поможет им уверенно работать с ней и эффективно настраивать под свои нужды, а также выполнять базовые операции после установки.

Содержание отчета

1. Наименование, цель занятия, дата;
2. Выполнение заданий;
3. Ответы на контрольные вопросы;
4. Вывод о проделанной работе.

📖 Контрольные вопросы:

1. Какие основные этапы включает процесс установки операционной системы?
2. Как подготовить загрузочный носитель для установки Windows?
3. Какие параметры необходимо настроить при первом запуске ОС?
4. Как установить обновления и драйверы после установки ОС?
5. Какие базовые операции необходимо выполнить после установки ОС?

6. Как создать точку восстановления системы?
7. Почему важно установить антивирусное программное обеспечение сразу после установки ОС?
8. Какие шаги необходимо выполнить для создания загрузочного носителя?
9. Какие системные требования бывают у различных операционных систем?
10. Опишите процесс установки ОС на виртуальной машине.
11. Почему важно устанавливать обновления после установки ОС?
12. Какие основные параметры безопасности нужно настроить после установки ОС?

Практическое занятие № 4

Тема: Работа с реестром конфигурационными файлами ОС

Цель занятия:

Обучиться основам работы с реестром Windows и конфигурационными файлами, понять их структуру, назначение и способы редактирования.

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

Реестр Windows — это централизованная база данных, содержащая настройки и параметры для операционной системы и установленных программ. Он состоит из ключей и значений, организованных в иерархическую структуру.

1. Структура реестра:
    - Главные разделы реестра:

- `HKEY_CLASSES_ROOT` — информация о зарегистрированных файловых форматах и COM-объектах.
- `HKEY_CURRENT_USER` — настройки текущего пользователя.
- `HKEY_LOCAL_MACHINE` — настройки для всей системы.
- `HKEY_USERS` — настройки для всех пользователей.
- `HKEY_CURRENT_CONFIG` — информация о текущем оборудовании.

2. Редактирование реестра:
    - Для редактирования используется утилита `regedit`.
    - Создание, изменение и удаление ключей и значений. Важно сохранять резервные копии перед внесением изменений.
3. Конфигурационные файлы:
    - Конфигурационные файлы часто имеют форматы `.ini`, `.cfg`, `.xml`, `.json` и используются для хранения настроек программ.
    - Общая структура: секции, параметры и значения.
4. Методы работы с конфигурационными файлами:
    - Чтение и запись с помощью текстовых редакторов (например, `Notepad`) или программных интерфейсов.

✍Задания для студентов

1. **Работа с редактором реестра**:
    - Откройте редактор реестра (regedit).
    - Перейдите в раздел **HKEY_CURRENT_USER\Software**.
    - Создайте новый ключ с именем **TestKey**.
    - Внутри ключа **TestKey** создайте строковое значение с именем **TestValue** и значением **Example**.
    - Экспортируйте ключ **TestKey** в файл **TestKey.reg**.
    - Удалите ключ **TestKey**.
    - Импортируйте ключ **TestKey** из файла **TestKey.reg**.
2. **Редактирование конфигурационных файлов**:
    - Создайте текстовый файл с именем **config.ini**.
    - Добавьте в файл следующие строки:

```
[Settings]
Parameter1=Value1
```

```
Parameter2=Value2
```

- Откройте файл **config.ini** и измените значение **Parameter1** на **NewValue1**.
- Сохраните изменения и закройте файл.

3. **Создание точки восстановления системы**:
   - Откройте "Панель управления" -> "Система и безопасность" -> "Система" -> "Защита системы".
   - Нажмите "Создать" и следуйте инструкциям на экране для создания точки восстановления.

Дополнительные задания для студентов

1. Изучение структуры реестра:
   - Откройте `regedit` и исследуйте основные разделы. Сделайте скриншоты ключей в каждом разделе.
2. Создание резервной копии реестра:
   - Сохраните резервную копию реестра (весь или только часть).
3. Редактирование реестра:
   - Добавьте новый ключ или значение (например, измените параметры автозагрузки программ).
4. Работа с конфигурационным файлом:
   - Найдите и измените конфигурационный файл одной из программ (например, измените настройки в `config.ini` или `settings.json` у простого приложения).
5. Создание собственного конфигурационного файла:
   - Создайте новый конфигурационный файл в формате `.ini` и напишите несколько секций и параметров (например, настройки для приложения).

Примеры команд

1. **Пример создания ключа и значения в реестре**:
   - Откройте редактор реестра (regedit).
   - Перейдите в раздел **HKEY_CURRENT_USER\Software**.
   - Щелкните правой кнопкой мыши на **Software** и выберите "Создать" -> "Раздел".
   - Назовите новый раздел **TestKey**.
   - Щелкните правой кнопкой мыши на **TestKey** и выберите "Создать" -> "Строковый параметр".
   - Назовите новый параметр **TestValue** и установите его значение на **Example**.
2. **Пример экспорта и импорта реестра**:
   - Щелкните правой кнопкой мыши на ключе **TestKey** и выберите "Экспортировать".
   - Сохраните файл как **TestKey.reg**.
   - Удалите ключ **TestKey**.
   - Дважды щелкните на файл **TestKey.reg** для импорта ключа обратно в реестр.
3. **Пример редактирования конфигурационного файла**:
   - Откройте текстовый редактор (например, Блокнот).
   - Создайте новый файл и добавьте следующие строки:
   - [Settings]

```
[Settings]
```

```
Store1= number1
Store2= number2
```

- Сохраните файл как **config.ini**.
- Откройте файл **config.ini** и измените значение **Store1** на **NewValue1**.
- Сохраните изменения и закройте файл.

Дополнительные примеры команд и действий:

1. Открытие реестра:
   - Ввод в командной строке:

```
regedit
```

2. Создание нового ключа в реестре:
   - В `HKEY_CURRENT_USER`, щелкните правой кнопкой мыши и выберите `Создать -> Ключ`, затем задайте имя ключа.
3. Редактирование конфигурационного файла:
   - Открытие файла `config.ini` в блокноте:

```
[Settings]
Volume=70
Fullscreen=True
```

4. Запись значения в реестре с помощью PowerShell:

```
Set-ItemProperty -Path 'HKCU:\Software\MyApp' -Name 'Setting1' -Value
'Value1'
```

По завершении занятия студенты научатся эффективно работать с реестром и конфигурационными файлами, получают навыки редактирования системных настроек и управления программным обеспечением, что будет полезно в их будущей практике работы с операционными системами.

🕮 Контрольные вопросы:

1. В чем разница между различными разделами реестра и их назначение?
2. Как создать резервную копию реестра и почему это важно?
3. Как можно отредактировать конфигурационный файл с помощью текстового редактора?
4. Какие опасности могут возникнуть при неправильном редактировании реестра?
5. Приведите примеры конфигурационных файлов и их назначение.
6. Что такое реестр Windows и зачем он нужен?
7. Какие основные разделы содержит реестр Windows?
8. Как создать, редактировать и удалить ключи и значения в реестре?
9. Как экспортировать и импортировать ключи реестра?
10. Что такое конфигурационные файлы и зачем они нужны?
11. Как редактировать конфигурационные файлы?
12. Почему важно создавать резервные копии реестра и конфигурационных файлов?
13. Как создать точку восстановления системы?

Практическое занятие № 5

Тема: Модели операционных систем. Ядро. Оболочка

Цель занятия:

Ознакомить студентов с различными моделями операционных систем и их архитектурой. Понять структуру и функциональность ядра операционной системы.Научиться различать типы ядер и их особенности. Познакомиться с основами работы с оболочкой операционной системы, научить использовать команды для управления файловой системой и процессами.

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

Оболочка операционной системы (от англ. shell «оболочка») — программа, предоставляющая интерфейс для взаимодействия пользователя с функциями системы.
Оболочка - это самый внешний уровень операционной системы. Оболочки содержат в себе язык программирования для управления процессами и файлами, а также запуска и управления другими программами. Оболочка управляет взаимодействием между вами и операционной системой, показывая вам приглашения для ввода, интерпретируя ввод для операционной системы, а затем обрабатывая результирующий вывод операционной системы.
Виды оболочек: командная (CLI) и графическая (GUI).
Примеры популярных оболочек: Bash, Zsh, PowerShell.

1. Модели операционных систем:
   - Монолитные ядра:
     - Все функции управления ресурсами, процессами и устройствами сгруппированы в одном коде.
     - Пример: Linux.
   - Микроядерные модели:
     - Минималистичное ядро, минимальный набор служб, остальные — в виде серверов в пользовательском пространстве.
     - Примеры: Minix, QNX.
   - Гибридные ядра:
     - Комбинация микроядерного и монолитного подходов. Содержит часть управляющего кода в ядре, а остальное в пользовательском пространстве.
     - Пример: Windows NT.
   - Виртуальные машины:
     - Эмулируют аппаратное обеспечение и позволяют запускать несколько операционных систем на одном физическом устройстве.
     - Пример: VMware, VirtualBox.
2. Ядро ОС:
   - Ядро — центральная часть операционной системы, отвечающая за управление аппаратными ресурсами (ЦП, память, устройства ввода-вывода).
   - Функции ядра:
     - Управление памятью
     - Управление процессами
     - Управление вводом/выводом
     - Обеспечение безопасности
3. Типы ядер:

- Монолитное ядро: Работает как единое целое, высокая производительность, но сложность в отладке.
- Микроядерное: Более высокая надежность и безопасность, но потенциально меньшая производительность.
- Гибридное: Попытка объединить преимущества монолитного и микроядерного подходов.

Основные команды оболочки:

- Навигация по файловой системе:
  - `ls` – список файлов и директорий.
  - `cd` – смена директории.
  - `pwd` – отображение текущей директории.
- Работа с файлами:
  - `cp` – копирование файлов.
  - `mv` – перемещение или переименование файлов.
  - `rm` – удаление файлов.
  - `touch` – создание новых файлов.
- Просмотр содержимого файлов:
  - `cat` – вывод содержимого файла.
  - `less` – постраничный просмотр файла.
- Поиск:
  - `find` – поиск файлов по именам.
  - `grep` – поиск по содержимому файлов.

3. Перенаправление и конвейеры:
   - Использование `>, >>, <` для перенаправления входных и выходных данных.
   - Конвейер `(|)` для передачи вывода одной команды во вход другой.
4. Скрипты и автоматизация:
   - Основы написания Bash-скриптов.
   - Синтаксис: переменные, условные конструкции, циклы.

✍Задания для студентов

1. Исследование моделей ОС:
   - Найдите три примера операционных систем для каждой из упомянутых моделей (монолитные, микроядерные, гибридные) и опишите их ключевые особенности.
2. Сравнительный анализ:
   - Составьте таблицу, сравнивающую производительность, надежность, безопасность и сложность разработки различных моделей ядер.
3. Практическое задание:
   - С использованием виртуальной машины, установите две различные операционные системы (например, Linux и Windows) и проведите эксперименты с производительностью, сравнив использование ресурсов в каждой из систем.
4. Обсуждение архитектуры ядра:
   - Опишите, как работает управление памятью в операционных системах на примере выбранной линии (например, Linux).

🎵 Примеры команд

- Монолитное ядро: Linux использует монолитную архитектуру, где все модули загружаются в одно пространство ядра.
- Микроядерное ядро: Minix разделяет функциональность, отделяя серверы управления от основного ядра.
- Гибридное ядро: Windows NT использует гибридный подход с некоторыми компонентами в режиме ядра для производительности и другими в пользовательском пространстве для стабильности.

Дополнительные задания для студентов

1. Навигация по файловой системе:
   - Перейдите в директорию `/home` и выведите список всех файлов и папок.
2. Работа с файлами:
   - Создайте текстовый файл `example.txt`, запишите туда несколько строк текста, затем скопируйте его в папку `/tmp`.
3. Просмотр содержимого файла:
   - Убедитесь, что содержимое файла `example.txt` правильное, используя команды `cat` и `less`.
4. Использование перенаправления:
   - Выведите список файлов в директории `/etc` в файл `etc_files.txt`.
5. Создание Bash-скрипта:
   - Напишите простой скрипт, который создает три файла (`file1.txt, file2.txt, file3.txt`) в текущей директории и выводит сообщение о создании.

Дополнительные примеры:

Пример команды для создания файла:

```
touch myfile.txt
```

Пример скрипта:

```
#!/bin/bash
echo "Создание файлов..."
touch file1.txt file2.txt file3.txt
echo "Файлы созданы."
```

Практическое занятие позволяет студентам освоить базовые команды и принципы работы с оболочкой операционной системы, что является основой для дальнейшего изучения более сложных тем в области операционных систем и администрирования, а также освоить ключевые концепции моделей операционных систем и архитектуры ядер.

🕮 Контрольные вопросы:

1. Что такое ядро операционной системы и его основные функции?

2. Каковы основные различия между монолитными, микроядерными и гибридными архитектурами яму?
3. Приведите примеры операционных систем для каждой из моделей.
4. Как микроядерная модель повышает безопасность системы?
5. Как управление памятью реализовано в современных операционных системах
6. Что такое оболочка и для чего она используется?
7. Каковы основные команды для работы с файлами в оболочке?
8. Как осуществляется перенаправление вывода команд?
9. Что такое конвейеры, и как они работают?
10. Какие элементы языка Bash нужно знать для написания простого скрипта?

Практическое занятие № 6

Тема: Управление процессами ОС Linux

Цель занятия:
Научиться основам управления процессами в ОС Linux, включая создание, мониторинг и завершение процессов, а также использование команд для управления задачами.

Краткие сведения из теории

Процесс — это выполняемая программа с определёнными ресурсами (память, процессор и пр.). Каждый процесс в Linux имеет уникальный идентификатор (PID).

1. Статусы процессов:
   - Запущенный **(Running)**
   - Ожидающий **(Waiting)**
   - Завершённый **(Stopped)**
   - Завершённый **(Terminated)**

2. Команды для управления процессами:
   - `ps` — отображение списка процессов.
     - Пример: `ps aux`— показать все процессы с информацией о пользователе и использовании ресурсов.
   - `top` — мониторинг активности процессов в реальном времени.
   - `htop` — улучшенный `top` интерактивный монитор процессов (не всегда установлен по умолчанию).
   - `kill` — завершение процесса по PID.
     - Пример: `kill 1234` — завершить процесс с PID 1234.
   - `killall` — завершение всех процессов с заданным именем.
     - Пример: `killall firefox` — завершить все запущенные экземпляры Firefox.
   - `bg` — перевод процесса между режимами.
   - `fg` — перевод процесса из фонового режима в передний план.
   - `jobs` — отображение списка фоновых задач.
3. Создание процессов:
   - Процессы создаются с помощью команды `fork()`, однако в командной строке можно запустить новый процесс с помощью команд.
4. Параллельные и фоновые процессы:
   - Запуск комад в фоне: добавление символа `&` в конце команды (например, `gedit &`).

Дополнительные команды для управления процессами:
- **pgrep**: Поиск процессов по имени или другим критериям.
- **pkill**: Отправка сигналов процессам, найденным с помощью **pgrep**.
- **nice** и **renice**: Управление приоритетами процессов.
- **&**: Запуск процесса в фоновом режиме.
- **jobs**: Просмотр фоновых и остановленных процессов.
- **fg** и **bg**: Перемещение процессов между фоном и передним планом.
- **nice** и **renice** команд для управления приоритетами

Сигналы процессов:
- Основные сигналы: **SIGTERM**, **SIGKILL**, **SIGSTOP**, **SIGCONT**.

✍Задания для студентов

1. **Просмотр списка процессов**:
   - Откройте терминал.
   - Используйте команду **ps** для просмотра списка запущенных процессов.
   - Используйте команду **ps aux** для просмотра подробной информации о процессах.
   - Используйте команду **top** для мониторинга процессов в реальном времени.
2. **Запуск и остановка процессов**:
   - Запустите процесс в фоновом режиме (например, **sleep 60 &**).
   - Используйте команду **jobs** для просмотра фоновых процессов.
   - Используйте команду **fg** для перемещения процесса на передний план.
   - Используйте команду **bg** для перемещения процесса в фоновый режим.
   - Используйте команду **kill** для завершения процесса (например, **kill <PID>**).
3. **Управление приоритетами процессов**:
   - Запустите процесс с низким приоритетом (например, **nice -n 10 sleep 60**).
   - Используйте команду **renice** для изменения приоритета запущенного процесса (например, **renice -n 5 <PID>**).
4. **Отправка сигналов процессам**:
   - Запустите процесс (например, **sleep 60**).
   - Используйте команду **kill** для отправки сигнала **SIGTERM** (например, **kill <PID>**).
   - Используйте команду **kill -9** для отправки сигнала **SIGKILL** (например, **kill -9 <PID>**).

♩ Примеры команд

1. **Пример команды для просмотра списка процессов**:

```
ps aux
```

2. **Пример команды для запуска процесса в фоновом режиме**:

```
sleep 60 &
```

3. **Пример команды для завершения процесса**:

```
kill <PID>
```

4. **Пример команды для изменения приоритета процесса**:

```
renice -n 5 <PID>
```

5. **Пример команды для отправки сигнала SIGKILL**:

```
kill -9 <PID>
```

Дополнительные задания для студентов

1. Запуск и мониторинг процессов:

- Запустите несколько экземпляров вашего любимого текстового редактора или терминала в фоне и используйте команду ps для их мониторинга.

2. Использование команды top:
   - Откройте top, просмотрите запущенные процессы, обратите внимание на использование ресурсов (CPU, RAM). Выведите информацию о процессах и сделайте снимок экрана.
3. Завершение процессов:
   - Найдите PID запущенного процесса с помощью `ps` и завершите его командой `kill`.
4. Работа с фоновыми процессами:
   - Запустите процесс в фоне, используя `&`, затем используйте команды `jobs`, `bg` и `fg`, чтобы контролировать и управлять фоновыми задачами.
5. Исследование команды `htop`:
   - Установите `htop` (если не установлен) и используйте его для мониторинга процессов. Попробуйте завершить один из процессов через htop.

Дополнительные примеры команд:

1. Просмотр всех процессов:

```
ps aux
```

2. Запуск `top`:

```
top
```

3. Завершение процесса по PID:

```
kill 1234
```

4. Завершение всех процессов с заданным именем:

```
killall firefox
```

5. Запуск процесса в фоне:

```
gedit &
```

В результате занятия студенты получат практические навыки управления процессами в Linux, научатся эффективно использовать системные команды для мониторинга и контроля за выполнением программ, что является важной частью администрирования ОС.

🕮 Контрольные вопросы:

1. Что такое процесс и какие его основные характеристики?
2. Какую информацию можно получить с помощью команды `ps`?
3. Как завершить процесс с помощью его PID?

4. В чем разница между командами `bg` и `fg`?
5. Какие команды используются для мониторинга процессов в реальном времени?
6. Что такое процесс и зачем он нужен?
7. Какие основные команды используются для просмотра списка процессов в Linux?
8. Как запустить процесс в фоновом режиме?
9. Как переместить процесс между фоном и передним планом?
10. Как завершить процесс с помощью команды **kill**?
11. Что такое приоритет процесса и как его изменить?
12. Какие основные сигналы существуют и как их отправить процессу?
13. Как использовать команду **top** для мониторинга процессов в реальном времени?

Практическое занятие № 7

Тема: Создание пользовательских скриптов ОС Unix

Цель занятия:
Научиться основам написания и выполнения пользовательских скриптов в операционных системах Unix, а также синтаксису и структуре скриптового языка оболочки (например, Bash).

Краткие сведения из теории

Скрипт — это набор команд, написанных в текстовом файле и выполняемых оболочкой Unix (например, Bash). Скрипты позволяют автоматизировать рутинные задачи, упрощая взаимодействие с системой. Основные интерпретаторы скриптов: Bash, Perl, Python.

1. Основы синтаксиса Bash:
    - Строки, начинающиеся с `#`, являются комментариями.
    - Переменные определяются без пробелов, присваивание: `VAR=value`.
    - Для доступа к значению переменной используется `$VAR`.
    - Условия и циклы:
        - Условное выражение: `if [ условие ]; then ... fi`.
        - Цикл `for`: `for i in {1..5}; do ... done`.
        - Цикл `while`: `while [ условие ]; do ... done`.
2. Создание и выполнение скриптов:
    - Скрипт создаётся в текстовом редакторе (например, `nano`, `vim`).
    - Необходимо добавить шебанг вначале файла: `#!/bin/bash`.
    - Для выполнения скрипта требуется изменить права доступа: `chmod +x script.sh`.
3. Основные команды и функции:
    - `echo` — вывод текста.
    - `read` — ввод данных от пользователя.
    - `exit` — завершение скрипта.
    - `function` — создание пользовательской функции.

Синтаксис и структура скриптов Bash:
- Шебанг (**#!/bin/bash**): указание интерпретатора.
- Переменные и их использование.
- Условные операторы (**if**, **else**, **elif**).
- Циклы (**for**, **while**, **until**).
- Функции.
- Ввод и вывод данных.

Основные команды и утилиты:
- **echo**: вывод текста на экран.
- **read**: чтение ввода пользователя.
- **grep**, **awk**, **sed**: обработка текста.
- **find**, **ls**, **cp**, **mv**, **rm**: работа с файлами и директориями.

Права доступа и выполнение скриптов:
- Установка прав доступа с помощью команды **chmod**.

- Выполнение скриптов.

Отладка и тестирование скриптов:

- Использование команды **bash -x** для отладки.
- Тестирование скриптов на различных данных.

✍Задания для студентов

1. **Создание простого скрипта**:
   - Создайте файл с именем **hello.sh**.
   - Добавьте в файл следующий код:

```
#!/bin/bash
echo "Hello, World!"
```

   - Установите права доступа для выполнения скрипта:

```
chmod +x hello.sh
```

   - Выполните скрипт:

```
./hello.sh
```

2. **Создание скрипта с переменными**:
   - Создайте файл с именем **variables.sh**.
   - Добавьте в файл следующий код:

```
#!/bin/bash
name="John"
echo "Hello, $name!"
```

   - Установите права доступа для выполнения скрипта и выполните его.

3. **Создание скрипта с условными операторами**:
   - Создайте файл с именем **conditions.sh**.
   - Добавьте в файл следующий код:

```
#!/bin/bash
read -p "Enter a number: " number
if [ $number -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is less than or equal to 10."
fi
```

   - Установите права доступа для выполнения скрипта и выполните его.

4. **Создание скрипта с циклом**:

- Создайте файл с именем **loop.sh**.
- Добавьте в файл следующий код:

```
#!/bin/bash
for i in {1..5}
do
    echo "Iteration $i"
done
```

- Установите права доступа для выполнения скрипта и выполните его.

5. **Создание скрипта для работы с файлами**:
   - Создайте файл с именем **file_operations.sh**.
   - Добавьте в файл следующий код:

```
#!/bin/bash
mkdir test_dir
touch test_dir/test_file.txt
echo "File created successfully."
```

- Установите права доступа для выполнения скрипта и выполните его.

Примеры команд:

1. **Пример простого скрипта**:

```
#!/bin/bash
echo "Hello, World!"
```

2. **Пример скрипта с переменными**:

```
#!/bin/bash
name="John"
echo "Hello, $name!"
```

3. **Пример скрипта с условными операторами**:

```
#!/bin/bash
read -p "Enter a number: " number
if [ $number -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is less than or equal to 10."
fi
```

4. **Пример скрипта с циклом**:

```
#!/bin/bash
```

```
for i in {1..5}
do
    echo "Iteration $i"
done
```

5. **Пример скрипта для работы с файлами**:

```
#!/bin/bash
mkdir test_dir
touch test_dir/test_file.txt
echo "File created successfully."
```

Дополнительные задания для студентов

1. Создание простого скрипта:
   - Напишите скрипт, который выводит "Hello, World!" на экран.
2. Ручной ввод:
   - Создайте скрипт, который запрашивает у пользователя его имя и выводит приветственное сообщение.
3. Условная конструкция:
   - Создайте скрипт, который проверяет, является ли введённое число четным или нечетным.
4. Цикл для вывода чисел:
   - Напишите скрипт, который выводит числа от 1 до 10 с помощью цикла `for`.
5. Работа с функциями:
   - Напишите скрипт с функцией, которая принимает два числа, выполняет сложение и выводит результат.

Дополнительные примеры команд:

1. Создание простого скрипта (**hello.sh**):

```
#!/bin/bash
echo "Hello, World!"
```

2. Скрипт с вводом имени (**greet.sh**):

```
#!/bin/bash
read -p "Введите ваше имя: " name
echo "Привет, $name!"
```

3. Скрипт для проверки четности (**check_even.sh**):

```
#!/bin/bash
read -p "Введите число: " number
if [ $((number % 2)) -eq 0 ]; then
    echo "$number является четным."
else
```

```
    echo "$number является нечетным."
fi
```

4. Скрипт для вывода чисел от 1 до 10 (**count.sh**):

```
#!/bin/bash
for i in {1..10}; do
    echo $i
done
```

5. Скрипт с функцией (**add.sh**):

```
#!/bin/bash
function add {
    echo $(($1 + $2))
}
read -p "Введите первое число: " num1
read -p "Введите второе число: " num2
sum=$(add $num1 $num2)
echo "Сумма: $sum"
```

Этот материал поможет студентам освоить основные принципы создания и использования пользовательских скриптов в операционной системе Unix, а также научит их писать и выполнять простые и сложные скрипты. По завершении занятия студенты смогут создавать простые скрипты в оболочке Unix, выполнять автоматизацию рутинных задач и использовать основные конструкции программы. Эти навыки помогут им в дальнейшем изучении системного администрирования и программирования в Unix-подобных системах.

🕮 Контрольные вопросы:

1. Что такое скрипт и зачем он нужен?
2. Какие основные интерпретаторы скриптов существуют в Unix?
3. Как указать интерпретатор в скрипте?
4. Как объявить и использовать переменные в скрипте Bash?
5. Какие условные операторы существуют в Bash?
6. Какие циклы существуют в Bash?
7. Как установить права доступа для выполнения скрипта?
8. Как выполнить скрипт в Unix?
9. Как отладить скрипт в Bash?
10. Какие команды используются для работы с файлами и директориями в скриптах?
11. Что такое скрипт и для чего он используется в Unix?
12. Какой синтаксис используется для создания комментариев в Bash?
13. Как задать условия и циклы в Bash?
14. Как изменить права доступа к скрипту для его выполнения?
15. Как можно передать аргументы функции в Bash?

Практическое занятие № 8

Тема: Настройка и работа с сетью.

Цель занятия:
Ознакомиться с основными принципами настройки и работы с сетью в операционной системе, научить их выполнять базовые операции по настройке сети и диагностике сетевых проблем.

Краткие сведения из теории

IP-адрес – уникальный адрес устройства в сети, используемый для идентификации и связи. Существует два типа:

- IPv4 (например, 192.168.1.1) — 32-битный адрес, позволяющий создать около 4 миллиардов уникальных адресов.
- IPv6 (например, 2001:0db8:85a3:0000:0000:8a2e:0370:7334) — 128-битный адрес, значительно увеличивающий доступное количество адресов.

Маска подсети – определяет, какая часть IP-адреса используется для идентификации сети, а какая — для идентификации устройства в сети. Например, маска 255.255.255.0 означает, что первые три октета адреса описывают сеть, а последний — устройства в ней.

Шлюз – устройство (обычно маршрутизатор или роутер – «домашний WiFi»), которое служит точкой доступа к другим сетям, обеспечивая связь между ними. Шлюз используется для отправки данных на устройства, находящиеся за пределами локальной сети.

DNS (Система доменных имен) – служба, которая преобразует удобочитаемые доменные имена (например, www.example.com) в IP-адреса, обеспечивая доступ к ресурсам в интернете. Позволяет пользователям не запоминать числовые адреса.

Типы сетей:

1. **Локальная сеть (LAN)**
    - Ограниченная географически сеть, которая объединяет устройства на небольшом расстоянии, например, в пределах одного здания или кампуса. Широко используется для обмена данными и ресурсами (принтеры, файлы) в организациях.
2. **Глобальная сеть (WAN)**
    - Сеть, которая охватывает большие географические области и соединяет множество локальных сетей. Примером WAN является интернет, позволяющий устройствам по всему миру связываться друг с другом.

Протоколы сетевого уровня

1. **TCP/IP (Transmission Control Protocol/Internet Protocol)**
    - Набор протоколов, обеспечивающий передачу данных в интернете. TCP отвечает за надежность и гарантирует доставку пакетов данных в правильном порядке, тогда как IP отвечает за адресацию и маршрутизацию.
2. **UDP (User Datagram Protocol)**
    - Протокол, предоставляющий невероятно быструю, но ненадежную передачу данных. Не ставит приоритет на доставку и порядок приходящих пакетов, что делает его подходящим для приложений, таких как видеотрансляции или онлайн-игры, где скорость важнее точности.

Настройка сети в операционной системе

1. Настройка IP-адреса, маски подсети, шлюза и DNS-серверов
a. IP-адрес
- Устанавливает уникальный адрес для устройства в сети.

b. Маска подсети
- Указывает, какая часть IP-адреса относится к сети, а какая — к устройству.

c. Шлюз
- Определяет, через какое устройство выходить в другие сети (например, в интернет).

d. DNS-серверы
- Позволяют преобразовывать доменные имена в IP-адреса для доступа к ресурсам.

Использование команд для настройки сети
a. Linux:
- **ifconfig**: Утилита для настройки сетевых интерфейсов (заменена на `ip` в новых системах).
    - Пример использования:

```
sudo ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

- **ip**: Более современная команда для работы с сетевыми интерфейсами.
    - Пример использования:

```
sudo ip addr add 192.168.1.10/24 dev eth0
sudo ip route add default via 192.168.1.1
```

b. Windows:
- netsh: Команда для сетевой настройки в Windows.
    - Пример использования:

```
netsh interface ip set address "Ethernet" static 192.168.1.10 255.255.255.0
192.168.1.1
netsh interface ip set dns "Ethernet" static 8.8.8.8
```

3. Настройка статического и динамического IP-адреса
a. Статический IP-адрес
- Установка фиксированного IP-адреса на устройство, что обеспечивает постоянное подключение с заданным адресом.
- Пример в Linux: В файле `/etc/network/interfaces`:

```
auto eth0
iface eth0 inet static
    address 192.168.1.10
    netmask 255.255.255.0
    gateway 192.168.1.1
    dns-nameservers 8.8.8.8 8.8.4.4
```

- Пример в Windows: В сетевых настройках устанавливаем вручную адрес и параметры.

b. Динамический IP-адрес

- Автоматическая установка IP-адреса с помощью DHCP-сервера. Устройство получает адрес, маску подсети, шлюз и DNS автоматически.
- Пример в Linux: В файле `/etc/network/interfaces`для DHCP:

```
auto eth0
iface eth0 inet dhcp
```

- Пример в Windows: В настройках сети выбираем "Получить IP-адрес автоматически".

Диагностика сетевых проблем

1. Использование команд для диагностики сетевых проблем

a. `ping`

- Команда используется для проверки доступности узла в сети. Отправляет ICMP-запросы и получает ответы, что позволяет определить, доступен ли удалённый хост.
    - Синтаксис:

```
ping <IP-адрес или доменное имя>
```

    - Пример:

```
ping www.example.com
```

    - Анализ результатов:
        - Если получают ответы, хост доступен.
        - Если нет ответов или появляется сообщение о превышении времени ожидания, возможно, проблемы с подключением или узел отключен.

b. `traceroute` (или `tracert` в Windows)

- Эта команда определяет путь пакетов к удалённому узлу, показывая промежуточные маршруты (хопы).
    - Синтаксис:

```
traceroute <IP-адрес или доменное имя>  # Linux
tracert <IP-адрес или доменное имя>     # Windows
```

    - Пример:

```
traceroute www.example.com
```

    - Анализ результатов:
        - Каждый хоп показывает, через какие узлы проходят пакеты, и их время отклика.

- Задержки (времена ответов) на каждом промежуточном узле могут указывать на проблемы с маршрутизацией или перегруженные узлы.

c. `netstat`

- Утилита для отображения сетевых соединений, маршрутов и статистики протоколов. Позволяет увидеть активные соединения.
    - Синтаксис:

```
netstat -an
```

    - Пример:

```
netstat -a | findstr ESTABLISHED  # Windows
```

    - Анализ результатов:
        - Просмотр активных соединений и их состояние (например, ESTABLISHED, CLOSED).
        - Может помочь определить проблемы с соединениями или приложения, использующие сетевые порты.

d. `nslookup`

- Команда для проверки DNS-записей. Используется для диагностики проблем с переводом доменных имен в IP-адреса.
    - Синтаксис:

```
nslookup <доменное имя>
```

    - Пример:

```
nslookup www.example.com
```

    - Анализ результатов:
        - Возвращает IP-адрес и информацию о DNS-сервере, который осуществил поиск.
        - Если запрос не удался, возможно, есть проблемы с DNS-сервером или сетью.

2. Анализ результатов диагностики

- Сеть доступна и функционирует:
    - Если команды `ping`, `traceroute`, и `nslookup` возвращают положительные результаты, значит, сеть работает без проблем.
- Проблемы с доступом к узлу:
    - Если `ping` не отвечает, а `traceroute` показывает задержки или пропуски на определенных хопах, это может указывать на проблемы с маршрутизацией или неправильной настройкой сети.
- Проблемы с DNS:
    - Если `nslookup` не возвращает IP-адрес для домена, необходимо проверить настройки DNS-сервера или работоспособность самого сервера.
- Занятые или закрытые порты:

- `netstat` помогает выявить, какие порты заняты и какие соединения активны. Если необходимо, можно выявить и закрыть ненужные соединения.

Безопасность сети

1. Основные принципы безопасности сети
   - Конфиденциальность – защита информации от несанкционированного доступа. Данные должны передаваться и храниться в зашифрованном виде.
   - Целостность – гарантия, что данные не были изменены или повреждены в процессе передачи. Используются криптографические хеши и контрольные суммы.
   - Доступность – обеспечение легкого и своевременного доступа к данным для авторизованных пользователей.
   - Аутентификация – подтверждение личности пользователей или устройств, чтобы гарантировать, что доступ к данным получают только авторизованные лица.
   - Аудит и мониторинг – регулярная проверка сетевых действий и использование журналов для выявления подозрительной активности.

2. Настройка брандмауэра

a. **Linux**

- **iptables**: Мощный инструмент для создания правил фильтрации трафика.
  - Примеры:
    - Разрешение входящих соединений только для SSH:

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

    - Блокировка всех входящих соединений, кроме разрешённых:

```
sudo iptables -A INPUT -j DROP
```

- **UFW (Uncomplicated Firewall)**: Упрощенный интерфейс для управления iptables.
  - Примеры:
    - Включение UFW:

```
sudo ufw enable
```

    - Разрешение порта 80 (HTTP):

```
sudo ufw allow 80
```

    - Проверка статуса:

```
sudo ufw status
```

b. **Windows Firewall**

- Встроенный брандмауэр в Windows, который можно настроить через Панель управления или PowerShell.
  - Примеры через **PowerShell**:
    - Разрешение приложения в файрволле:

```
New-NetFirewallRule -DisplayName "MyApp" -Direction Inbound -Program
"C:\Path\To\MyApp.exe" -Action Allow
```

- Блокировка входящего трафика на определённый порт:

```
New-NetFirewallRule -DisplayName "Block Port 1234" -Direction Inbound -
LocalPort 1234 -Protocol TCP -Action Block
```

3. Использование виртуальной частной сети для защиты данных

- виртуальная частная сеть – защищённый туннель для передачи данных между устройствами через общедоступные сети. виртуальной частной сети шифрует трафик, что обеспечивает конфиденциальность и безопасность данных в интернете.
- Польза использования виртуальной частной сети:
  - Защита данных при использовании общедоступных Wi-Fi сетей (например, в кафе или аэропортах), изменение IP-адреса.
  - Защита от шпионского ПО и перехвата данных.
- Выбор виртуальной частной сети:
  - Выбирайте надёжные службы виртуальной частной сети с хорошими отзывами и безопасной политикой использования данных.
  - Обратите внимание на поддержку протоколов шифрования (например, Open, IKEv2).
  - Проверьте наличие функций защиты, таких как "аварийное отключение" и "защита от утечек DNS".

Работа с сетевыми утилитами

1. Использование утилит для сканирования сети

a. **Nmap (Network Mapper)**

- Описание: **Nmap** — мощная утилита для сканирования сетей, позволяющая выявлять активные устройства, открытые порты, службы и операционные системы.
- Установка:
  - Linux:

```
sudo apt install nmap
```

  - Windows: Скачайте и установите с официального сайта.
- Основные команды **Nmap**:
  - Сканирование всех устройств в подсети:

```
nmap 192.168.1.0/24
```

  - Сканирование конкретного хоста для выявления открытых портов:

```
nmap 192.168.1.10
```

  - Определение операционной системы:

```
nmap -O 192.168.1.10
```

  - Aggressive scan (агрессивное сканирование):

```
nmap -A 192.168.1.10
```

- Анализ результатов:

- Nmap предоставляет информацию об открытых портах, службах, версиях и операционных системах. Это может помочь в обнаружении уязвимых мест сети и потенциальных угроз.

2. Использование утилит для мониторинга сети

a. **Wireshark**

- Описание: Wireshark — популярный инструмент для анализа сетевого трафика, позволяющий захватывать и исследовать пакеты данных, проходящие через сеть.
- Установка:
  - Linux:

```
sudo apt install wireshark
```

  - Windows: Скачайте и установите с официального сайта.
- Основные функции **Wireshark**:
  - Запись трафика: Начните захват пакетов, выбрав интерфейс сети.
  - Фильтрация пакетов: Используйте фильтры для отображения конкретного трафика, например, TCP или HTTP:
    - Пример фильтра:
      - Для отображения только HTTP-трафика:

```
http
```

- Анализ данных:
  - Просмотр заголовков пакетов, данных и протоколов.
  - Возможность отслеживания сессий для анализа взаимодействия между хостами.
- Обнаружение проблем:
  - Идентификация узких мест, высоких задержек и ненадежных соединений по анализу временных задержек и повторяющихся пакетов.

✍Задания для студентов

1. **Настройка статического IP-адреса**:
   - Откройте терминал или командную строку.
   - Настройте статический IP-адрес для сетевого интерфейса (например, **ifconfig eth0 192.168.1.100 netmask 255.255.255.0** в Linux или **netsh interface ip set address "Local Area Connection" static 192.168.1.100 255.255.255.0 192.168.1.1** в Windows).
   - Проверьте настройки сети (например, **ifconfig** в Linux или **ipconfig** в Windows).
2. **Диагностика сетевых проблем**:
   - Используйте команду **ping** для проверки доступности удаленного хоста (например, **ping google.com**).
   - Используйте команду **traceroute** для определения маршрута до удаленного хоста (например, **traceroute google.com** в Linux или **tracert google.com** в Windows).
   - Используйте команду **netstat** для просмотра активных сетевых соединений (например, **netstat -tuln** в Linux или **netstat -ano** в Windows).

- Используйте команду **nslookup** для проверки DNS-записей (например, **nslookup google.com**).

3. **Настройка брандмауэра**:
   - Настройте брандмауэр для разрешения или запрета определенных портов (например, **ufw allow 80/tcp** в Linux или **netsh advfirewall firewall add rule name="Allow HTTP" protocol=TCP dir=in localport=80 action=allow** в Windows).
   - Проверьте настройки брандмауэра (например, **ufw status** в Linux или **netsh advfirewall firewall show rule name=all** в Windows).
4. **Использование сетевых утилит**:
   - Установите и запустите утилиту **nmap** для сканирования сети (например, **nmap 192.168.1.0/24**).
   - Установите и запустите утилиту **wireshark** для мониторинга сетевого трафика.

Примеры команд:

1. **Пример команды для настройки статического IP-адреса в Linux**:

sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0

2. **Пример команды для настройки статического IP-адреса в Windows**:

netsh interface ip set address "Local Area Connection" static 192.168.1.100 255.255.255.0 192.168.1.1

3. **Пример команды для проверки доступности удаленного хоста**:

ping google.com

4. **Пример команды для определения маршрута до удаленного хоста**:

traceroute google.com

5. **Пример команды для просмотра активных сетевых соединений**:

netstat -tuln

6. **Пример команды для проверки DNS-записей**:

nslookup google.com

7. **Пример команды для настройки брандмауэра в Linux**:

sudo ufw allow 80/tcp

8. **Пример команды для настройки брандмауэра в Windows**:

netsh advfirewall firewall add rule name="Allow HTTP" protocol=TCP dir=in localport=80 action=allow

Дополнительные задания для студентов

Задание 1: Настройка сетевого интерфейса

1. Откройте терминал.
2. Используя команду `ip a`, просмотрите доступные сетевые интерфейсы.
3. Настройте статический IP-адрес для выбранного интерфейса и добавьте шлюз.

Задание 2: Работа с DNS

1. Проверьте текущие настройки DNS, используя `/etc/resolv.conf`.
2. Измените настройки DNS на альтернативный DNS-сервер (например, Google DNS — 8.8.8.8).
3. Проверьте работоспособность новых настроек с помощью команды `nslookup`.

Задание 3: Диагностика соединения

1. Используйте команду `ping` для проверки доступности локального и удалённого сервера.

2. Примените `traceroute` для диагностики маршрута до удалённого сервера.

Дополнительные примеры команд

1. Просмотр сетевых интерфейсов:

```
ip a
```

2. Настройка IP-адреса:

```
sudo ip addr add 192.168.0.10/24 dev eth0
```

3. Установка шлюза:

```
sudo ip route add default via 192.168.0.1
```

4. Проверка DNS:

```
nslookup www.example.com
```

5. Пинг до сервера:

```
ping 8.8.8.8
```

🕮 Контрольные вопросы:

1. Что такое IP-адрес, маска подсети, шлюз и DNS?
2. Какие основные команды используются для настройки сети в Linux и Windows?
3. Как настроить статический IP-адрес в Linux и Windows?
4. Какие команды используются для диагностики сетевых проблем?
5. Как использовать команду **ping** для проверки доступности удаленного хоста?
6. Как использовать команду **traceroute** для определения маршрута до удаленного хоста?
7. Как использовать команду **netstat** для просмотра активных сетевых соединений?
8. Как использовать команду **nslookup** для проверки DNS-записей?
9. Как настроить брандмауэр в Linux и Windows?
10. Какие утилиты используются для сканирования и мониторинга сети?
11. Какова структура IP-адреса?
12. В чем разница между статическим и динамическим IP-адресами?
13. Как работает DNS и почему он необходим?
14. Какие команды можно использовать для диагностики сетевых соединений?
15. Что такое маршрутизация и как она влияет на сетевую связь?

Практическое занятие № 9

Тема: Основные принципы безопасности.

Цель занятия:
Ознакомиться с основными понятиями и принципами безопасности информации. Рассмотреть классификацию угроз и методы защиты. Изучить технологии и механизмы обеспечения безопасности в современных системах.

Краткие сведения из теории

Основные понятия безопасности:
- Конфиденциальность: защита данных от несанкционированного доступа.
- Целостность: защита данных от несанкционированного изменения, обеспечение правильности и полноты информации
- Доступность: обеспечение доступности ресурсов и данных для авторизованных пользователей.
- Аутентификация: процесс подтверждения личности пользователя.
- Авторизация: процесс предоставления доступа к ресурсам на основе прав пользователя.
- Информационная безопасность: Защита информации от несанкционированного доступа, использования, раскрытия, разрушения или изменения.

Классификация угроз:
- Вирусы и черви: вредоносное ПО, распространяющееся самостоятельно.
- Трояны: вредоносное ПО, маскирующееся под полезные программы.
- Фишинг: мошеннические попытки получить конфиденциальную информацию.
- Атаки типа "отказ в обслуживании" (DoS): атаки, направленные на перегрузку системы.
- Уязвимости программного обеспечения: ошибки в коде, которые могут быть использованы для атаки.
- Угрозы внешнего характера: Вирусы, хакеры, несанкционированный доступ.
- Внутренние угрозы: Ошибки пользователей, злонамеренные действия сотрудников.
- Естественные угрозы: Пожары, наводнения, землетрясения.

Базовые технологии безопасности:
- Антивирусное ПО: программы для обнаружения и удаления вредоносного ПО.
- Брандмауэры: программы для контроля сетевого трафика.
- Шифрование: методы защиты данных от несанкционированного доступа.
- VPN: технологии для создания защищенных соединений.
- Аутентификация: Процессы проверки подлинности пользователей (пароль, биометрия, токены).

Механизмы защиты:
- Политики безопасности: наборы правил и процедур для обеспечения безопасности.

- Резервное копирование: создание копий данных для восстановления в случае потери. (регулярное создание копий данных для защиты от их потери).
- Обновления и патчи: регулярное обновление программного обеспечения для устранения уязвимостей.
- Мониторинг и аудит: отслеживание и анализ событий безопасности.
- Контроль доступа: политики, определяющие, кто имеет доступ к системе или информации.
- Системы обнаружения вторжений (IDS): мониторинг сетевого трафика на предмет подозрительной активности.

Надежные системы:
- Отказоустойчивость: способность системы продолжать работу при сбоях.
- Избыточность: использование дублирующих компонентов для повышения надежности.
- Балансировка нагрузки: распределение нагрузки между несколькими серверами для повышения производительности и надежности.
- Разработка систем с учетом безопасности как основного принципа.
- Использование архитектурных подходов, таких как многослойная безопасность и изоляция компонентов.

✍Задания для студентов

1. **Установка и настройка антивирусного ПО**:
    - Установите антивирусное ПО (например, ClamAV в Linux или антивирус в Windows).
    - Настройте регулярное сканирование системы на наличие вирусов.
    - Проверьте систему на наличие вирусов.
2. **Настройка брандмауэра**:
    - Настройте брандмауэр для разрешения или запрета определенных портов (например, **sudo ufw allow 80/tcp** в Linux или **netsh advfirewall firewall add rule name="Allow HTTP" protocol=TCP dir=in localport=80 action=allow** в Windows).
    - Проверьте настройки брандмауэра (например, **sudo ufw status** в Linux или **netsh advfirewall firewall show rule name=all** в Windows).
3. **Шифрование данных**:
    - Установите и настройте утилиту для шифрования данных (например, GPG в Linux или BitLocker в Windows).
    - Зашифруйте файл и проверьте, что он не может быть прочитан без ключа.
4. **Резервное копирование данных**:
    - Настройте регулярное резервное копирование данных (например, с помощью утилиты **rsync** в Linux или встроенных инструментов в Windows).
    - Проверьте, что резервные копии создаются и могут быть восстановлены.

Примеры команд:

1. **Пример установки и настройки антивирусного ПО в Linux**:

```
sudo apt-get install clamav
sudo freshclam
sudo clamscan -r /
```

2. **Пример настройки брандмауэра в Linux**:

```
sudo ufw allow 80/tcp
sudo ufw status
```

3. **Пример шифрования данных с помощью GPG в Linux**:

```
gpg --gen-key
gpg --output doc.gpg --encrypt --recipient user@example.com doc
```

4. **Пример настройки VPN с помощью OpenVPN в Linux**:

```
sudo apt-get install openvpn
sudo openvpn --config /path/to/config.ovpn
```

5. **Пример резервного копирования данных с помощью rsync в Linux**:

```
rsync -avz /source/directory /backup/directory
```

Самостоятельные задания для студентов

Задание 1: Исследование угроз

1. Выберите одну из классификаций угроз (внешние, внутренние, естественные).
2. Подготовьте краткий доклад о выбранной угрозе, включая примеры инцидентов и возможные последствия.

Задание 2: Разработка политики безопасности

1. Составьте предложение по политике аутентификации для организации. Укажите:
   - Минимальные требования к паролям.
   - Процесс восстановления пароля.
   - Механизмы двухфакторной аутентификации (если применимо).

Задание 3: Анализ механизма защиты

1. Выберите один из механизмов защиты (брандмауэр, IDS, контроль доступа).
2. Подготовьте презентацию с описанием его принципов работы, преимуществ и недостатков.

Примеры технологий.

Пример 1: Шифрование

- **AES (Advanced Encryption Standard)**: Общепринятый стандарт для шифрования данных на уровне файлов или сообщений.

Пример 2: Аутентификация

- Двухфакторная аутентификация (**2FA**): Комбинация пароля и SMS-кода для подтверждения личности пользователя.

Пример 3: Брандмауэр

- **iptables**: Утилита в Linux для настройки правил фильтрации трафика.

🕮 Контрольные вопросы:

1. Что такое конфиденциальность, целостность и доступность в контексте безопасности?
2. Какие основные угрозы существуют для информационных систем?
3. Какие базовые технологии безопасности используются для защиты систем?
4. Как настроить антивирусное ПО в Linux и Windows?
5. Как настроить брандмауэр в Linux и Windows?
6. Как использовать шифрование для защиты данных?
7. Как настроить VPN для защиты сетевого трафика?
8. Как настроить регулярное резервное копирование данных?
9. Какие механизмы защиты используются для обеспечения безопасности систем?
10. Что такое надежные системы и какие принципы используются для их создания?
11. Какие основные понятия информционной безопасности вы знаете?
12. Как классифицируются угрозы для информационных систем?
13. Что такое контроль доступа и как он может быть реализован?
14. Какие меры помогут разработать надежную систему безопасности?

Практическое занятие № 10

Тема: Резервное копирование и восстановление данных в Windows.

Цель занятия:
Ознакомиться с основными принципами резервного копирования и восстановления данных в операционной системе Windows, научиться использовать встроенные и сторонние инструменты для создания и восстановления резервных копий.

Краткие сведения из теории

1. Важность резервного копирования
   - Защита данных: Обеспечение безопасности важной информации от потери из-за сбоев системы, удаления или вирусных атак.
   - Восстановление после сбоев: Позволяет восстановить систему и данные в случаи потери.
2. Основные методы резервного копирования
   - Полное резервное копирование: Копирование всех данных без исключения.
   - Инкрементное резервное копирование: Копируются только измененные данные с момента последнего резервного копирования.
   - Дифференциальное резервное копирование: Копируются все данные, измененные с последнего полного резервного копирования.
3. Встроенные инструменты Windows для резервного копирования
   - История файлов: Позволяет автоматически создавать резервные копии файлов и восстановить их при необходимости.
   - Резервное копирование и восстановление: Для создания образа системы и резервного копирования данных.
   - Windows Backup: Использование планировщика для автоматизации задач резервного копирования.
   - Точка восстановления системы (System Restore): восстановление системы до предыдущего состояния.

Дополнительная информация

1. **Основные понятия резервного копирования**:

Резервное копирование — это процесс создания копий данных, файлов или систем для защиты информации от потери, повреждения или удаления. Оно необходимо для восстановления данных в случае их утраты из-за сбоев, атак или других непредвиденных обстоятельств.
Основные компоненты резервного копирования: данные, системные файлы, приложения.

2. **Сторонние инструменты для резервного копирования**:
   - Acronis True Image: мощный инструмент для создания резервных копий и восстановления данных.
   - Macrium Reflect: инструмент для создания образов дисков и резервных копий.
   - Veeam Backup: решение для резервного копирования и восстановления виртуальных машин.
3. **Процесс создания резервной копии**:
   - Выбор данных для резервного копирования.
   - Настройка расписания резервного копирования.
   - Выбор места хранения резервных копий (локальный диск, внешний диск, облако).

4. **Процесс восстановления данных**:
    - Выбор резервной копии для восстановления.
    - Восстановление данных на исходное место или в другое место.
    - Проверка целостности восстановленных данных.

✍Задания для студентов

Задание 1: Настройка Истории файлов

1. Откройте "Настройки" > "Обновление и безопасность" > "Резервное копирование".
2. Настройте Историю файлов для резервного копирования личных файлов на внешний диск.
3. Проверьте, как осуществляется автоматическое резервное копирование.

Задание 2: Создание образа системы

1. Перейдите в Панель управления и выберите "Резервное копирование и восстановление (Windows 7)".
2. Создайте полный образ системы на внешнем носителе.
3. Запишите, какие параметры были выбраны для резервного копирования.

Задание 3: Восстановление данных

1. На основе созданного образа системы осуществите восстановление файлов (можно использовать тестовые файлы).
2. Зафиксируйте процесс восстановления и результаты.

♪ Примеры команд:

Пример 1: Настройка Истории файлов

- Включение функции:
    - Подключите внешний диск, откройте "Настройки" > "Резервное копирование".
    - Нажмите "Добавить диск" и выберите подключенный диск.

Пример 2: Создание образа системы

- Запуск резервного копирования через Панель управления:
    - В "Резервное копирование и восстановление" выберите "Создать образ системы" и следуйте инструкциям.

Пример 3: Восстановление с помощью образа системы

- В случае сбоя компьютера, используя восстановительный диск, зайдите в "Восстановление системы" и выберите восстановление из созданного образа.

Дополниетельные задания для студентов:

1. **Настройка истории файлов**:
    - Откройте "Панель управления" -> "Система и безопасность" -> "История файлов".
    - Настройте истории файлов для резервного копирования файлов пользователя на внешний диск.
    - Проверьте, что резервные копии создаются автоматически.
2. **Создание резервной копии системы**:
    - Откройте "Панель управления" -> "Система и безопасность" -> "Резервное копирование и восстановление".
    - Настройте резервное копирование системы на внешний диск.
    - Создайте резервную копию системы.
3. **Создание точки восстановления системы**:
    - Откройте "Панель управления" -> "Система и безопасность" -> "Система" -> "Защита системы".

- Создайте точку восстановления системы.
- Проверьте, что точка восстановления создана успешно.

4. **Использование стороннего инструмента для резервного копирования**:
   - Установите и настройте Acronis True Image или Macrium Reflect.
   - Создайте резервную копию системы или данных с помощью стороннего инструмента.
   - Проверьте, что резервная копия создана успешно.
5. **Восстановление данных из резервной копии**:
   - Восстановите данные из резервной копии, созданной с помощью истории файлов или стороннего инструмента.
   - Проверьте целостность восстановленных данных.

Примеры:

1. **Пример настройки истории файлов**:
   - Откройте "Панель управления" -> "Система и безопасность" -> "История файлов".
   - Нажмите "Включить" и выберите внешний диск для хранения резервных копий.
   - Нажмите "Включить" для завершения настройки.
2. **Пример создания резервной копии системы**:
   - Откройте "Панель управления" -> "Система и безопасность" -> "Резервное копирование и восстановление".
   - Нажмите "Настройка резервного копирования" и выберите внешний диск для хранения резервных копий.
   - Выберите данные для резервного копирования и нажмите "Далее".
   - Нажмите "Сохранить настройки и запустить резервное копирование".
3. **Пример создания точки восстановления системы**:
   - Откройте "Панель управления" -> "Система и безопасность" -> "Система" -> "Защита системы".
   - Нажмите "Создать" и введите описание точки восстановления.
   - Нажмите "Создать" для завершения процесса.
4. **Пример использования стороннего инструмента для резервного копирования**:
   - Установите Acronis True Image или Macrium Reflect.
   - Откройте программу и выберите опцию создания резервной копии.
   - Выберите данные для резервного копирования и место хранения.
   - Нажмите "Создать резервную копию" для завершения процесса.
5. **Пример восстановления данных из резервной копии**:
   - Откройте истории файлов или сторонний инструмент.
   - Выберите резервную копию для восстановления.
   - Выберите данные для восстановления и место восстановления.
   - Нажмите "Восстановить" для завершения процесса.

🕮 Контрольные вопросы:

1. Что такое резервное копирование и зачем оно нужно?
2. Какие типы резервного копирования существуют?
3. Какие встроенные инструменты Windows используются для резервного копирования?
4. Как настроить историю файлов в Windows?
5. Как создать резервную копию системы в Windows?
6. Как создать точку восстановления системы в Windows?

7. Какие сторонние инструменты используются для резервного копирования?
8. Как создать резервную копию с помощью стороннего инструмента?
9. Как восстановить данные из резервной копии?
10. Как проверить целостность восстановленных данных?
11. Какова основная цель резервного копирования данных?
12. Какие методы резервного копирования вы знаете?
13. Что означает инкрементное резервное копирование?
14. В чем заключается процесс восстановления данных из резервной копии?
15. Чем отличается Дифференциальное от Полного резервного копирования?

Практическое занятие № 11

Тема: Резервное копирование и восстановление данных в Unix.

Цель занятия:
Ознакомиться с основными принципами резервного копирования и восстановления данных в операционной системе Unix, научиться использовать встроенные и сторонние инструменты для создания и восстановления резервных копий.

Краткие сведения из теории

1. Важность резервного копирования в Unix

- Защита данных от потери из-за аппаратных сбоев, повреждений файлов или ошибок пользователей.
- Возможность оперативного восстановления системы до рабочего состояния.

2. Основные методы резервного копирования

- Полное резервное копирование: Копирование всех файлов и данных.
- Инкрементное резервное копирование: Копирование только новых или измененных файлов с момента последнего резервного.
- Дифференциальное резервное копирование: Копирование всех файлов, измененных с последнего полного резервного копирования.

3. Инструменты для резервного копирования в Unix

- **tar**: Утилита для создания архивов.
- **rsync**: Позволяет выполнять резервное копирование и синхронизацию данных.
- **dump/restore**: Предназначен для резервного копирования и восстановления файловых систем.
- **cp**: Простая команда для копирования файлов и директорий.

Основные понятия резервного копирования:

- Что такое резервное копирование и зачем оно нужно.
- Типы резервного копирования: полное, инкрементное, дифференциальное.
- Основные компоненты резервного копирования: данные, системные файлы, приложения.

Встроенные инструменты Unix для резервного копирования:

- **dd**: создание образов дисков и разделов.
- **cron**: планирование задач для автоматического резервного копирования.

Сторонние инструменты для резервного копирования:

- **rsnapshot**: инструмент для создания резервных копий с использованием **rsync**.
- **Bacula**: мощное решение для резервного копирования и восстановления данных.
- **Duplicity**: инструмент для создания резервных копий с использованием шифрования.

1. **Процесс создания резервной копии**:
    - Выбор данных для резервного копирования.

- Настройка расписания резервного копирования.
- Выбор места хранения резервных копий (локальный диск, внешний диск, облако).

2. **Процесс восстановления данных**:
   - Выбор резервной копии для восстановления.
   - Восстановление данных на исходное место или в другое место.
   - Проверка целостности восстановленных данных.

✍Задания для студентов

1. **Создание резервной копии с помощью tar**:
   - Откройте терминал.
   - Создайте архив резервной копии директории (например, **/home/user/documents**):

```
tar -czvf backup.tar.gz /home/user/documents
```

   - Проверьте, что архив создан успешно.
2. **Синхронизация файлов с помощью rsync**:
   - Откройте терминал.
   - Синхронизируйте директорию с внешним диском (например, **/mnt/backup**):

```
rsync -avz /home/user/documents /mnt/backup
```

   - Проверьте, что файлы синхронизированы успешно.
3. **Создание образа диска с помощью dd**:
   - Откройте терминал.
   - Создайте образ диска (например, **/dev/sda**):

```
sudo dd if=/dev/sda of=disk_image.img
```

   - Проверьте, что образ диска создан успешно.
4. **Настройка автоматического резервного копирования с помощью cron**:
   - Откройте терминал.
   - Откройте файл crontab для редактирования:

```
crontab -e
```

   - Добавьте задачу для автоматического резервного копирования (например, каждый день в 2 часа ночи):

```
0 2 * * * tar -czvf /mnt/backup/backup.tar.gz /home/user/documents
```

   - Сохраните изменения и закройте редактор.
5. **Восстановление данных из резервной копии**:
   - Откройте терминал.
   - Восстановите данные из архива резервной копии:

```
tar -xzvf backup.tar.gz -C /home/user/documents
```

- Проверьте целостность восстановленных данных.

Дополнительные задания для студентов

Задание 1: Создание полного резервного копирования с помощью tar

1. Запустите терминал и создайте каталог для резервной копии, если его нет:

```
mkdir ~/backup
```

2. Используйте команду tar для создания архива домашней директории:

```
tar -cvzf ~/backup/home_backup.tar.gz ~/
```

Задание 2: Инкрементное резервное копирование с **rsync**

1. Создайте директорию для последующих резервных копий:

```
mkdir ~/incremental_backup
```

2. Выполните первую полную резервную копию:

```
rsync -a --delete ~/ ~/incremental_backup/
```

3. Измените или добавьте несколько файлов, а затем выполните инкрементное резервное копирование:

```
rsync -a --delete ~/ ~/incremental_backup/
```

Задание 3: Восстановление данных

1. Восстановите файлы из архива, созданного с помощью tar:

```
tar -xvzf ~/backup/home_backup.tar.gz -C /путь_к_восстановлению
```

2. Проверьте, что файлы были восстановлены корректно.

🕮 Контрольные вопросы:

1. Что такое резервное копирование и зачем оно нужно?
2. Какие типы резервного копирования существуют?
3. Какие встроенные инструменты Unix используются для резервного копирования?
4. Как создать резервную копию с помощью **tar**?
5. Как синхронизировать файлы с помощью **rsync**?
6. Как создать образ диска с помощью **dd**?

7. Как настроить автоматическое резервное копирование с помощью **cron**?
8. Какие сторонние инструменты используются для резервного копирования в Unix?
9. Как восстановить данные из резервной копии?
10. Как проверить целостность восстановленных данных?
11. Каковы основные принципы и цели резервного копирования данных в Unix?
12. Чем отличается инкрементное резервное копирование от дифференциального?
13. Каковы основные инструменты, используемые для резервного копирования в Unix, и в чем их особенности?
14. Как восстановить файлы из созданного резервного архива?

Практическое занятие № 12

Тема: Настройка сетевого протокола TPC/IP в Unix.

Цель занятия:
Ознакомиться с основами настройки протоколов TCP и IP в операционных системах.

Краткие сведения из теории

1. Основы **TCP/IP**
   - **TCP (Transmission Control Protocol)**: Протокол уровня транспорта, обеспечивающий надежную передачу данных с контролем ошибок и порядком передачи.
   - **IP (Internet Protocol)**: Протокол сетевого уровня, отвечающий за адресацию и маршрутизацию пакетов данных в сети.
2. Структура **IP**-адреса
   - **IPv4**: 32-битный адрес, обычно представляемый в десятичном формате, разбитом на четыре октета (например, 192.168.1.37).
   - **IPv6**: 128-битный адрес, предназначенный для решения проблемы исчерпания адресов IPv4.
3. Команды для настройки **TCP/IP**
   - **ifconfig**: Используется для настройки сетевых интерфейсов (в большинстве дистрибутивов Linux).
   - **ip**: Более современная утилита для управления сетевыми интерфейсами и маршрутами.
   - **netstat**: Показывает сетевые соединения, маршруты и другие сетевые параметры.
   - **ping**: Проверяет доступность узлов в сети.
   - **traceroute**: Отслеживает маршрут пакетов к заданному узлу.

✍Задания для студентов

Задание 1: Настройка IP-адреса

1. Откройте терминал.
2. Используя команду `ip`, настройте статический IP-адрес для вашего сетевого интерфейса (например, `eth0`):

```
sudo ip addr add 192.168.1.100/24 dev eth0
sudo ip link set eth0 up
```

Задание 2: Настройка шлюза

1. Добавьте шлюз для доступа в другие сети:

```
sudo ip route add default via 192.168.1.1
```

2. Проверьте таблицу маршрутизации с помощью команды:

```
ip route show
```

Задание 3: Проверка соединения

1. Проверьте доступность локального узла или внешнего (например, Yandex):

```
ping 77.88.8.8
```

2. Используйте traceroute для отслеживания маршрута к Yandex:

```
traceroute yandex.ru
```

🕮 Контрольные вопросы:

1. Какова основная функция протокола TCP в стеке TCP/IP?
2. Что представляет собой IP-адрес и как он структурирован?
3. Какую команду вы используете для отображения сетевых интерфейсов и их конфигурации?
4. Чем команда ping отличается от traceroute?
5. Как проверить текущую таблицу маршрутизации в системе?
6. Что такое TCP/IP и зачем он нужен?
7. Какие основные компоненты TCP/IP существуют?
8. Какие команды используются для настройки сети в Unix?
9. Как настроить статический IP-адрес в Unix?
10. Как настроить DNS-серверы в Unix?
11. Какие команды используются для диагностики сетевых проблем?
12. Как использовать команду ping для проверки доступности удаленного хоста?
13. Как использовать команду traceroute для определения маршрута до удаленного хоста?
14. Как использовать команду netstat для просмотра активных сетевых соединений?
15. Как использовать команду nslookup для проверки DNS-записей?
16. Как настроить сетевые интерфейсы с помощью файлов конфигурации?
17. Как настроить брандмауэр в Unix?

Практическое занятие № 13

Тема: Настройка сетевого протокола FTP, SSH.

Цель занятия:
Ознакомиться с основными принципами настройки и использования сетевых протоколов FTP и SSH в операционных системах Windows и Unix, научиться выполнять базовые операции по настройке и использованию этих протоколов.

Краткие сведения из теории

1. Протокол FTP (File Transfer Protocol)

- FTP используется для передачи файлов между клиентом и сервером. Работает по модели клиент-сервер, использует порты 21 (команды) и 20 (данные).
- Режимы работы:
  - Активный режим: Сервер инициирует соединение с клиентом.
  - Пассивный режим: Клиент инициирует оба соединения.

2. Протокол SSH (Secure Shell)

- SSH предоставляет защищённый доступ к удалённым системам. Использует шифрование для защиты передаваемых данных.
- Основные команды: `ssh`, `scp` (для копирования файлов), `sftp` (SSH File Transfer Protocol).

3. Настройка FTP-сервера

- Windows: Использование IIS для настройки FTP.
- Unix: Установка и настройка **vsftpd** или **proftpd**.

4. Настройка SSH-сервера

- Windows: Установка OpenSSH через "Управление Windows".
- Unix: Обычно устанавливается и настраивается по умолчанию.

Дополнительная информация

1. **Введение в FTP**:
   - Что такое FTP и зачем он нужен.
   - Основные команды FTP: **get**, **put**, **ls**, **cd**, **mkdir**, **rmdir**, **delete**.
   - Использование FTP-клиентов (например, FileZilla).
2. **Настройка FTP-сервера**:
   - Установка и настройка FTP-сервера в Windows (например, FileZilla Server).
   - Установка и настройка FTP-сервера в Unix (например, vsftpd).
   - Настройка пользователей и прав доступа.
3. **Введение в SSH**:
   - Что такое SSH и зачем он нужен.
   - Основные команды SSH: **ssh**, **scp**, **sftp**.
   - Использование SSH-клиентов (например, PuTTY в Windows, OpenSSH в Unix).
4. **Настройка SSH-сервера**:

- Установка и настройка SSH-сервера в Windows (например, OpenSSH).
- Установка и настройка SSH-сервера в Unix (например, OpenSSH).
- Настройка пользователей и ключей доступа.

5. **Безопасность FTP и SSH**:
   - Основные принципы безопасности FTP и SSH.
   - Настройка брандмауэра для FTP и SSH.
   - Использование шифрования для защиты данных.

✍**Задания для студентов**

1. **Установка и настройка FTP-сервера в Windows**:
   - Скачайте и установите FileZilla Server.
   - Настройте пользователей и права доступа.
   - Проверьте доступ к FTP-серверу с помощью FTP-клиента (например, FileZilla).
2. **Установка и настройка FTP-сервера в Unix**:
   - Установите vsftpd:

```
sudo apt-get install vsftpd
```

   - Настройте конфигурационный файл **/etc/vsftpd.conf**.
   - Перезапустите сервер vsftpd:

```
sudo systemctl restart vsftpd
```

   - Проверьте доступ к FTP-серверу с помощью FTP-клиента (например, FileZilla).
3. **Установка и настройка SSH-сервера в Windows**:
   - Установите OpenSSH Server:

```
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
```

   - Настройте конфигурационный файл **C:\ProgramData\ssh\sshd_config**.
   - Перезапустите сервер OpenSSH:

```
Start-Service sshd
```

   - Проверьте доступ к SSH-серверу с помощью SSH-клиента (например, PuTTY).
4. **Установка и настройка SSH-сервера в Unix**:
   - Установите OpenSSH Server:

```
sudo apt-get install openssh-server
```

   - Настройте конфигурационный файл **/etc/ssh/sshd_config**.
   - Перезапустите сервер OpenSSH:

```
sudo systemctl restart ssh
```

   - Проверьте доступ к SSH-серверу с помощью SSH-клиента (например, OpenSSH).
5. **Настройка безопасности FTP и SSH**:
   - Настройте брандмауэр для разрешения доступа к FTP и SSH (например, **sudo ufw allow 21/tcp** для FTP и **sudo ufw allow 22/tcp** для SSH в Unix).
   - Настройте шифрование для защиты данных (например, использование SFTP вместо FTP).

🎵 **Примеры команд:**

1. **Пример установки и настройки FTP-сервера в Unix**:

```
sudo apt-get install vsftpd
sudo nano /etc/vsftpd.conf
sudo systemctl restart vsftpd
```

2. **Пример установки и настройки SSH-сервера в Unix**:

```
sudo apt-get install openssh-server
sudo nano /etc/ssh/sshd_config
sudo systemctl restart ssh
```

3. **Пример установки и настройки FTP-сервера в Windows**:
    - Скачайте и установите FileZilla Server.
    - Настройте пользователей и права доступа через интерфейс FileZilla Server.
4. **Пример установки и настройки SSH-сервера в Windows**:

```
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
Start-Service sshd
```

5. **Пример настройки брандмауэра для FTP и SSH в Unix**:

```
sudo ufw allow 21/tcp
sudo ufw allow 22/tcp
```

Дополнительные задания для студентов

Задание 1: Настройка FTP-сервера в Windows

1. Убедитесь, что IIS установлен на вашем компьютере.
2. Запустите "Диспетчер IIS" и выберите "Добавить сайт FTP".
3. Настройте анонимный доступ и выберите папку для хранилища.

Задание 2: Настройка SSH-сервера в Windows

1. Установите OpenSSH через "Приложения и возможности" в Windows.
2. Включите OpenSSH-сервер:

```
Get-Service -Name sshd | Set-Service -StartupType 'Automatic'
Start-Service sshd
```

Задание 3: Установка и настройка FTP-сервера в Unix

1. Установите vsftpd:

```
sudo apt update
sudo apt install vsftpd
```

2. Отредактируйте конфигурационный файл /etc/vsftpd.conf, включая:
    - Разрешение анонимного доступа:

```
anonymous_enable=YES
```

3. Перезапустите службу:

```
sudo systemctl restart vsftpd
```

Задание 4: Использование FTP и SSH-клиентов

1. Подключитесь к вашему FTP-серверу с помощью командной строки или клиента (например, FileZilla).
2. Используйте команду ftp в терминале для подключения к FTP-серверу.
3. Используйте SSH для подключения к удаленному серверу:

```
ssh user@hostname
```

4. Скопируйте файл с помощью SCP:

```
scp filename user@hostname:/path/to/destination
```

Дополнительные примеры

Пример 1: Подключение к FTP-серверу через командную строку

```
ftp ftp://localhost
```

Пример 2: Подключение к SSH-серверу

```
ssh user@localhost
```
Пример 3: Копирование файла с SCP
```
scp localfile.txt user@remote_host:/remote/directory/
```
Контрольные вопросы

🕮 Контрольные вопросы:

1. Что такое FTP и зачем он нужен?
2. Какие основные команды используются в FTP?
3. Как установить и настроить FTP-сервер в Windows?
4. Как установить и настроить FTP-сервер в Unix?
5. Что такое SSH и зачем он нужен?
6. Какие основные команды используются в SSH?
7. Как установить и настроить SSH-сервер в Windows?
8. Как установить и настроить SSH-сервер в Unix?
9. Какие основные принципы безопасности FTP и SSH существуют?
10. Как настроить брандмауэр для FTP и SSH?
11. Как использовать шифрование для защиты данных при использовании FTP и SSH?
12. Каковы основные функции протокола FTP?
13. В чем различие между активным и пассивным режимами FTP?
14. Как SSH обеспечивает безопасность?
15. Какова основная команда для подключения к SSH-серверу?
16. Какой протокол используется для защищенной передачи файлов?

Практическое занятие № 14

Тема: Настройка проводного доступа к сети

Цель занятия:
Знакомство с основами настройки проводного сетевого соединения.

Перечень используемого оборудования:
Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

Проводной доступ к сети обеспечивает стабильное и высокоскоростное подключение к интернету и локальным сетям. Настройка проводного доступа включает в себя подключение кабеля Ethernet к сетевому устройству и настройку сетевых параметров на компьютере.

1. Основы компьютерных сетей:
    - Сетевые топологии: шина, звезда, кольцо.
    - Протоколы: TCP/IP, DHCP, DNS.
    - Основные компоненты сети: коммутаторы, маршрутизаторы, сетевые адаптеры.
    - Ethernet: Технология для передачи данных по кабелю.
    - IP-адрес: Уникальный адрес, присваиваемый устройству в сети.
    - DHCP: Протокол для автоматического назначения IP-адресов.
    - DNS: Система доменных имен, преобразующая доменные имена в IP-адреса.
    - Шлюз: Устройство, соединяющее различные сети
2. Настройка сетевого соединения:
    - Подключение кабеля Ethernet: Вставьте кабель Ethernet в сетевой порт компьютера и в сетевое устройство (роутер, коммутатор).
    - Параметры сети: IP-адрес, маска подсети, шлюз, DNS-сервер.
    - DHCP vs. статическая конфигурация: принципы работы и использование.
3. Операционные системы:
    - Процессы настройки сетевого адаптера в Windows и Linux.
    - Возможности командной строки: `ipconfig`, `ifconfig`, `ping`, `traceroute`.

✍**Задания для студентов**

1. Настройка адаптера в Windows:
    - Откройте "Центр управления сетями и общим доступом".
    - Выберите "Изменение параметров адаптера", выберите проводной адаптер.
    - Настройте TCP/IP, используя статический IP-адрес.
2. Настройка адаптера в Linux:
    - Используйте команду `ifconfig` для просмотра текущих настроек.
    - Настройте сетевой интерфейс с помощью команды `ip` или редактирования файла конфигурации.

- Для настройки статического IP-адреса отредактируйте файл `/etc/network/interfaces` (для Debian-based дистрибутивов) или используйте NetworkManager.
-
- Пример конфигурации для статического IP-адреса:

```
auto eth0
iface eth0 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    gateway 192.168.1.1
    dns-nameservers 8.8.8.8
```

3. Проверка подключения:
    - Используйте команду `ping` для проверки связи с локальным маршрутизатором.
    - Проверьте доступность интернет-ресурсов.

Примеры команд:

1. Windows:
    - Статическая настройка TCP/IP:
        1. Ввести IP-адрес: `192.168.1.10`.
        2. Маска подсети: `255.255.255.0`.
        3. Шлюз: `192.168.1.1`.
        4. DNS сервер: `8.8.8.8`.
2. Linux:
    - Команда для настройки:

```
sudo ip addr add 192.168.1.10/24 dev eth0
sudo ip route add default via 192.168.1.1
```

    - Перезапустите сетевой интерфейс командой **sudo ifdown eth0 && sudo ifup eth0**.

Контрольные вопросы:

1. Что такое DHCP и как он работает?
2. Каковы основные различия между статическим и динамическим IP-адресом?
3. Какие команды можно использовать для диагностики сетевых проблем в Windows и Linux?
4. Почему важно правильно настраивать маску подсети?
5. онтрольные вопросы
6. Что такое Ethernet и для чего он используется?
7. Какие параметры необходимо настроить для подключения к сети?
8. Как настроить автоматическое получение IP-адреса на Windows?
9. Как настроить статический IP-адрес на Linux?

Практическое занятие № 15

Тема: Настройка беспроводного доступа к сети

Цель занятия:

Знакомство с основами работы с языком программирования, научиться писать простые программы и понять основные концепции программирования.

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

Беспроводной доступ к сети (Wi-Fi) позволяет устройствам подключаться к интернету и локальным сетям без использования проводов. Настройка беспроводного доступа включает в себя подключение к сети Wi-Fi и настройку сетевых параметров на компьютере.

1. Основы беспроводных сетей:
    - Определение WLAN и его компоненты.
    - Протоколы: IEEE 802.11 (a/b/g/n/ac/ax).
    - Шифрование и безопасность: WEP, WPA, WPA2, WPA3.
2. Параметры сетевого подключения:
    - SSID (имя сети), тип шифрования, пароль для доступа.
    - IP-адресация в беспроводных сетях: DHCP и статическая настройка.
3. Операционные системы:
    - Процесс подключения к беспроводной сети в Windows и Linux.
    - Инструменты для диагностики: `ping`, `iwconfig`, `nmcli`.

Основные понятия

1. **Wi-Fi**: Технология беспроводной передачи данных.
2. **SSID**: Идентификатор сети (имя сети).
3. **Пароль**: Код доступа к сети.
4. **IP-адрес**: Уникальный адрес, присваиваемый устройству в сети.
5. **DHCP**: Протокол для автоматического назначения IP-адресов.
6. **DNS**: Система доменных имен, преобразующая доменные имена в IP-адреса.
7. **Шлюз**: Устройство, соединяющее различные сети.

✍**Задания для студентов**

1. Подключение к беспроводной сети в Windows:
    - Открыть "Настройки сети и интернета".
    - Выбрать "Wi-Fi", найти доступные сети.
    - Ввести пароль для подключения и настроить параметры IP, если необходимо.
2. Подключение к беспроводной сети в Linux:
    - Использовать команду nmcli для поиска доступных сетей:

```
nmcli device wifi list
```

    - Подключиться к сети с помощью команды:

```
nmcli device wifi connect "SSID" password "ваш_пароль"
```

3. Проверка подключения:
    - Использовать команду `ping` для проверки доступности интернет-ресурсов.

🎼 Примеры команд:

1. Windows:
    - Подключение к сети:
        1. Нажмите на значок Wi-Fi в панели задач.
        2. Выберите сеть, введите пароль.
        3. Убедитесь, что подключение успешно установлено.
2. Linux:
    - Использование команды для подключения:

```
nmcli device wifi connect "Home_Network" password "my_password"
```

    - Проверка статус сетевого интерфейса:

```
nmcli connection show
```

🕮 Контрольные вопросы:

1. Каковы основные протоколы беспроводных сетей и их отличия?
2. Что такое SSID и почему он важен для подключения к сети?
3. Какие методы шифрования используются для защиты беспроводных сетей?
4. Как можно проверить качество сигнала беспроводного подключения в Linux?
5. Что такое Wi-Fi и для чего он используется?
6. Какие параметры необходимо настроить для подключения к сети Wi-Fi?
7. Как настроить автоматическое получение IP-адреса на Windows?
8. Как настроить статический IP-адрес на Linux?

Практическое занятие № 16

Тема: Понятие хостинга

Цель занятия:

Знакомство с основами хостинга и его типами. Изучение принципов работы хостинговых услуг и их роли в развертывании веб-приложений. Практическое применение знаний о хостинге

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

Хостинг — это услуга предоставления ресурсов для размещения веб-сайтов, приложений и других данных на серверах, доступных через интернет. Хостинг-провайдеры предоставляют необходимые ресурсы, такие как дисковое пространство, оперативная память, процессорное время и сетевые ресурсы.

1. Определение хостинга:
    - Хостинг — это услуга, предоставляющая пространство на сервере для размещения веб-сайтов и приложений.
    - Хостинг обеспечивает доступ к данным через интернет.
2. Типы хостинга:
    - Виртуальный хостинг: Один сервер обслуживает несколько сайтов. Подходит для небольших проектов.
    - Выделенный сервер: Полный контроль над сервером, подходит для крупных проектов с высоким трафиком.
    - *VPS* (Virtual Private Server): Разделение одного физического сервера на несколько виртуальных, обеспечивающее большую гибкость по сравнению с виртуальным хостингом.
    - Облачный хостинг: Ресурсы распределяются между несколькими серверами, обеспечивая высокую доступность и масштабируемость.
    - Хостинг для приложений: Специализированные услуги для работы с конкретными технологиями и языками программирования.
3. Ключевые характеристики хостинга:
    - Производительность: Скорость загрузки и обработка запросов.
    - Надежность: Время безотказной работы сервиса.
    - Безопасность: Поддержка SSL, защиты от DDoS-атак и управление бэкапами.
    - Техническая поддержка: Уровень и доступность поддержки пользователей.

Основные понятия

1. **Хостинг**: Услуга предоставления ресурсов для размещения веб-сайтов.
2. **Хостинг-провайдер**: Компания, предоставляющая услуги хостинга.
3. **Сервер**: Компьютер, на котором размещаются веб-сайты и приложения.
4. **Доменное имя**: Уникальное имя, используемое для доступа к веб-сайту.

5. **DNS**: Система доменных имен, преобразующая доменные имена в IP-адреса.
6. **FTP**: Протокол передачи файлов, используемый для загрузки файлов на сервер.

Виды хостинга

1. **Виртуальный хостинг (Shared Hosting)**:
    - Несколько веб-сайтов размещаются на одном сервере.
    - Ресурсы сервера делятся между всеми пользователями.
    - Подходит для небольших и средних веб-сайтов.
2. **Виртуальный выделенный сервер (VPS)**:
    - Виртуальный сервер с выделенными ресурсами.
    - Подходит для веб-сайтов с умеренным трафиком и требующих больше контроля.
3. **Выделенный сервер (Dedicated Server)**:
    - Физический сервер, полностью выделенный для одного пользователя.
    - Подходит для крупных веб-сайтов и приложений с высоким трафиком.
4. **Облачный хостинг (Cloud Hosting)**:
    - Хостинг, использующий облачные технологии для масштабирования ресурсов.
    - Подходит для веб-сайтов с переменным трафиком.
5. **Реселлерский хостинг (Reseller Hosting)**:
    - Хостинг, предоставляемый перепродавцам для дальнейшей перепродажи.
    - Подходит для веб-дизайнеров и агентств.

Выбор хостинга

1. **Определите требования**: Оцените объем трафика, необходимое дисковое пространство и другие ресурсы.
2. **Сравните провайдеров**: Изучите отзывы, сравните цены и услуги.
3. **Проверьте поддержку**: Убедитесь, что провайдер предоставляет круглосуточную поддержку.
4. **Тестируйте скорость и надежность**: Проверьте время отклика и надежность серверов.

✍Задания для студентов

1. Изучение различных типов хостинга:
    - Найдите и проанализируйте три различных хостинг-провайдера, сравнив их предложения по виртуальному, VPS и облачному хостингу.
2. Создание простого веб-сайта:
    - Используя бесплатный хостинг (например, GitHub Pages или Netlify), создайте простой веб-сайт с минимальным содержанием (HTML/CSS).
3. Подготовка презентации:
    - Разработайте презентацию на тему "Преимущества и недостатки различных типов хостинга".

Дополнительные задания для студентов

Задание 1: Изучите несколько хостинг-провайдеров и сравните их услуги и цены.
Задание 2: Зарегистрируйте доменное имя и подключите его к хостингу.
Задание 3: Загрузите файлы веб-сайта на сервер с помощью FTP.
Задание 4: Настройте базу данных на сервере и подключите ее к веб-сайту.

♪ Примеры команд:

1. Обзор хостинг-провайдеров:
    - Bluehost: Популярный виртуальный хостинг для малых бизнесов, предлагает 24/7 поддержку и множество инструментов.
    - DigitalOcean: Отличный для VPS с простым управлением и хорошими тарифами для разработчиков.
    - Amazon Web Services (AWS): Лидер на облачном рынке, предлагающий масштабируемые решения для крупных приложений.
2. Подключение к панеле управления:
    - Создайте учетную запись у провайдера хостинга, выполните вход в панель управления и ознакомьтесь с интерфейсом.

Примеры работы с хостингом

1. **Пример 1**: Сравнение хостинг-провайдеров.
    - Создайте таблицу с колонками: Провайдер, Цена, Дисковое пространство, Оперативная память, Поддержка, Отзывы.
    - Заполните таблицу данными для нескольких провайдеров.
2. **Пример 2**: Регистрация доменного имени.
    - Перейдите на сайт регистратора доменов (например, GoDaddy, Namecheap).
    - Выберите доменное имя и проверьте его доступность.
    - Заполните форму регистрации и оплатите услугу.
3. **Пример 3**: Загрузка файлов на сервер с помощью FTP.
    - Установите FTP-клиент (например, FileZilla).
    - Подключитесь к серверу, используя предоставленные хостинг-провайдером данные (хост, логин, пароль).
    - Перетащите файлы веб-сайта в соответствующую директорию на сервере.

🕮 Контрольные вопросы:

1. Какие типы хостинга существуют и в чем их основные различия?
2. Как выбор типа хостинга влияет на производительность веб-сайта?
3. Какие факторы следует учитывать при выборе хостинг-провайдера?
4. Что такое облачный хостинг и в каких случаях он предпочтительнее других типов?
5. Что такое хостинг и для чего он используется?
6. Какие виды хостинга существуют и в чем их особенности?
7. Как выбрать хостинг-провайдера?
8. Как зарегистрировать доменное имя?
9. Как загрузить файлы на сервер с помощью FTP?
10. Как настроить базу данных на сервере?

Практическое занятие № 17

Тема: Обзор серверных дистрибутивов операционных систем

Цель занятия:
Знакомство с основными серверными дистрибутивами операционных систем.

Перечень используемого оборудования:
Монитор, системный блок, устройства ввода-вывода.

Краткие сведения из теории

Серверные дистрибутивы операционных систем предназначены для использования на серверах, обеспечивая стабильность, безопасность и высокую производительность. Они используются для размещения веб-сайтов, баз данных, приложений и других сервисов.

1. Определение серверного дистрибутива:
    - Серверные дистрибутивы — это специализированные версии операционных систем, оптимизированные для работы на серверном оборудовании, обеспечивающие высокую производительность и стабильность.
2. Основные серверные дистрибутивы:
    - Ubuntu Server:
        - Популярный дистрибутив на базе Debian.
        - Легкость в установке и использовании, подходит для начинающих.
        - Поддержка LTS (Long Term Support) версий.
    - CentOS:
        - На основе Red Hat Enterprise Linux (RHEL).
        - Стабильная и безопасная среда для серверов.
        - Обширное сообщество и поддержка.
    - Debian:
        - Надежный и универсальный дистрибутив, используемый для серверов.
        - Известен своей стабильностью и широким выбором пакетов.
    - Fedora Server:
        - Ближайший родственник RHEL, предлагает новые технологии.
        - Быстрое обновление пакетов, предназначен для разработчиков.
    - SUSE Linux Enterprise Server (SLES):
        - Коммерческий дистрибутив, предлагающий поддержку и решение для бизнеса.
        - Отличается хорошей интеграцией с enterprise-решениями.
    - Arch Linux:
        - Подходит для опытных пользователей, позволяя настраивать систему под свои нужды.
        - Модель "каталога" (rolling release) обеспечивает постоянные обновления.

Основные понятия

1. **Серверная операционная система**: ОС, оптимизированная для работы на серверах.
2. **Дистрибутив**: Версия операционной системы, включающая определенный набор программного обеспечения и настроек.
3. **Стабильность**: Способность системы работать без сбоев и перезагрузок.

4. **Безопасность**: Защита системы от несанкционированного доступа и атак.
5. **Производительность**: Способность системы эффективно использовать ресурсы.

Дополнительная информация по популярным серверным дистрибутивам

1. **Ubuntu Server**
    - Основана на Debian.
    - Поддержка LTS (Long Term Support) версий.
    - Легкость установки и использования.
    - Подходит для различных задач, включая веб-серверы, базы данных и облачные сервисы.
2. **CentOS**
    - Клон Red Hat Enterprise Linux (RHEL).
    - Поддержка LTS версий.
    - Широко используется в корпоративной среде.
    - Подходит для крупных и средних предприятий.
3. **Debian**
    - Один из старейших и наиболее стабильных дистрибутивов.
    - Огромное количество пакетов.
    - Подходит для различных задач, включая веб-серверы, базы данных и научные вычисления.
4. **Red Hat Enterprise Linux (RHEL)**
    - Коммерческий дистрибутив.
    - Высокая стабильность и поддержка.
    - Широко используется в корпоративной среде.
    - Подходит для крупных предприятий.
5. **SUSE Linux Enterprise Server (SLES)**
    - Коммерческий дистрибутив.
    - Высокая стабильность и поддержка.
    - Подходит для крупных предприятий и облачных сервисов.
6. **FreeBSD**
    - Операционная система на основе BSD.
    - Высокая стабильность и производительность.
    - Подходит для веб-серверов, баз данных и сетевых устройств.

Критерии выбора серверного дистрибутива

1. **Стабильность и надежность**: Важность для критически важных систем.
2. **Поддержка и обновления**: Наличие долгосрочной поддержки и регулярных обновлений.
3. **Совместимость с программным обеспечением**: Поддержка необходимых приложений и сервисов.
4. **Сообщество и документация**: Наличие активного сообщества и качественной документации.
5. **Лицензирование и стоимость**: Коммерческие и бесплатные варианты.

✍Задания для студентов

1. Сравнительный анализ:
    - Выберите три серверных дистрибутива и составьте сравнительную таблицу по следующим параметрам: легкость установки, поддержка пакетов, стабильность, сообщество и поддержка.
2. Установка тестового дистрибутива:
    - Скачайте ISO образ одного из дистрибутивов (например, Ubuntu Server).

- Установите его в виртуальной машине (например, используя VirtualBox или VMware) и настройте основные параметры.

3. Создание краткого отчета:
    - После установки подготовьте отчет, включающий: процесс установки, основные настройки и преимущества использования данного дистрибутива.

♪ Примеры команд:

1. Сравнительная таблица:

Дистрибутив	Легкость установки	Поддержка пакетов	Стабильность	Сообщество
Ubuntu Server	Высокая	APT	Высокая	Широкое, много документации
CentOS	Средняя	YUM	Очень высокая	Активное сообщество
Debian	Средняя	APT	Высокая	Много лет на рынке

2. Процесс установки Ubuntu Server:
    - Запустите виртуальную машину с загрузкой по ISO файлу.
    - Выберите язык установки и следуйте указаниям установщика.
    - Настройте сети и создайте учетную запись администратора.

Дополнительные задания для студентов

1. **Задание 1**: Изучите официальные сайты Ubuntu Server, CentOS и Debian. Сравните их особенности и преимущества.
2. **Задание 2**: Установите Ubuntu Server на виртуальную машину. Настройте базовые параметры системы.
3. **Задание 3**: Установите CentOS на виртуальную машину. Настройте базовые параметры системы.
4. **Задание 4**: Сравните процесс установки и настройки Ubuntu Server и CentOS. Напишите отчет.

Примеры выполнения

1. **Пример 1**: Установка Ubuntu Server.
    - Скачайте ISO-образ Ubuntu Server с официального сайта.
    - Создайте виртуальную машину в VirtualBox или VMware.
    - Загрузитесь с ISO-образа и следуйте инструкциям установщика.
    - Настройте базовые параметры системы, такие как имя пользователя, пароль и сетевые настройки.
2. **Пример 2**: Установка CentOS.
    - Скачайте ISO-образ CentOS с официального сайта.
    - Создайте виртуальную машину в VirtualBox или VMware.
    - Загрузитесь с ISO-образа и следуйте инструкциям установщика.
    - Настройте базовые параметры системы, такие как имя пользователя, пароль и сетевые настройки.

🕮 Контрольные вопросы:

1. Каковы основные различия между серверными дистрибутивами Linux?
2. В каких случаях предпочтительнее использовать Debian по сравнению с CentOS?
3. Каково значение LTS версий и почему они важны для серверной эксплуатации?
4. Какие преимущества предлагает Fedora Server для разработчиков?

5. Что такое серверная операционная система и для чего она используется?
6. Какие популярные серверные дистрибутивы существуют и в чем их особенности?
7. Какие критерии следует учитывать при выборе серверного дистрибутива?
8. Как установить Ubuntu Server на виртуальную машину?
9. Как установить CentOS на виртуальную машину?
10. Какие преимущества и недостатки имеют различные серверные дистрибутивы?

Список используемой литературы

Основная

1. Гостев, И. М. Операционные системы : учебник и практикум для среднего профессионального образования / И. М. Гостев. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2024. — 164 с. — (Профессиональное образование). — ISBN 978-5-534-04951-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/539078 (дата обращения: 04.09.2024).
2. Назаров С.В. Современные операционные системы [Электронный ресурс] / С.В. Назаров, А.И. Широков. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2020. — 351 c. — 978-5-9963-0416-5. — Режим доступа: http://www.iprbookshop.ru/52176.html
3. Шаньгин В.Ф. Информационная безопасность и защита информации [Электронный ресурс] / В.Ф. Шаньгин. — Электрон. текстовые данные. — Саратов: Профобразование, 2017. — 702 c. — 978-5-4488-0070-2. — Режим доступа: http://www.iprbookshop.ru/63594.html
4. Сафонов В.О. Основы современных операционных систем [Электронный ресурс] / В.О. Сафонов. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 826 c. — 978-5-9963-0495-0.— Режим доступа: http://www.iprbookshop.ru/62818.html
5. Украинский, А.В. Методическое пособие по выполнению практических занятий при изучении дисциплины «Операционные системы и среды», – ТТЖТ, 2024.
6. Украинский, А.В. Методическое пособие по выполнению самостоятельной работы при изучении дисциплины «Операционные системы и среды», – ТТЖТ, 2024.

Интернет-ресурсы

www.ttgt.org (Сайт Тихорецкого Техникума Железнодорожного Транспорта)

www.studentlibrary.ru (Электронная библиотека)

www.biblio-online.ru (Электронная библиотека)

www.fcior.edu.ru (Федеральный центр информационно-образовательных ресурсов — ФЦИОР).

www.school-collection. edu. ru (Единая коллекция цифровых образовательных ресурсов).

www.intuit. ru/studies/courses (Открытые интернет-курсы «Интуит» по курсу «Информатика»).

www. lms. iite. unesco. org (Открытые электронные курсы «ИИТО ЮНЕСКО» по информационным технологиям).

http://ru. iite. unesco. org/publications (Открытая электронная библиотека «ИИТО ЮНЕСКО» по ИКТ в образовании).