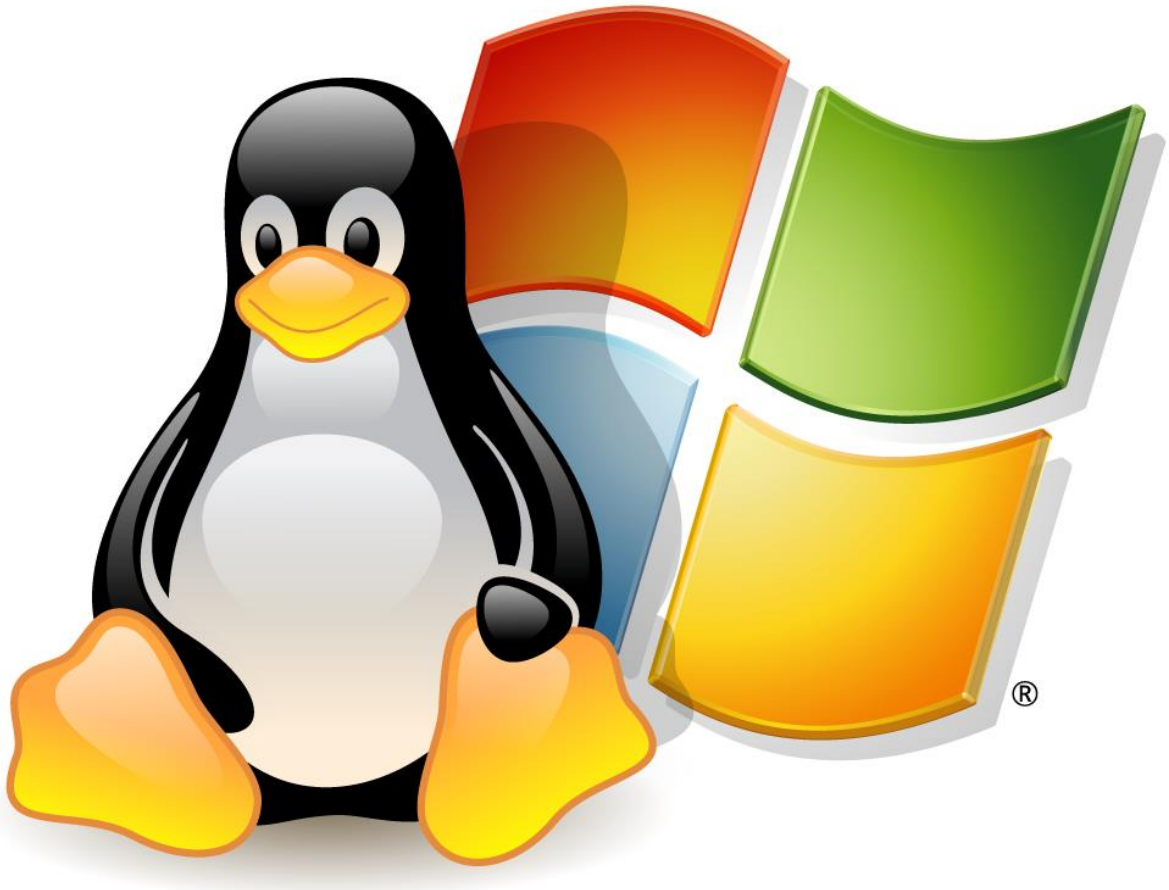
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Д.О. Пахмурин

ОПЕРАЦИОННЫЕ СИСТЕМЫ ЭВМ

Учебно-методическое пособие к лабораторным работам



Томск – 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра промышленной электроники

Д.О. Пахмурин

ОПЕРАЦИОННЫЕ СИСТЕМЫ ЭВМ

Учебно-методическое пособие к лабораторным работам для студентов очной формы обучения по направлению 11.03.04 – Электроника и наноэлектроника (профиль "Промышленная электроника")

2016

Пахмурин Д.О.
Операционные системы ЭВМ: Учебно-методическое пособие к лабораторным работам. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2016. – 76 с.

Приведены методические указания для выполнения лабораторных работ по дисциплине "Операционные системы ЭВМ", определена тематика и порядок их выполнения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5
1. Работа с файлами и дисками в ОС Windows XP 6
1.1. Краткие теоретические сведения 6
1.2. Подготовка к выполнению лабораторной работы 9
1.3. Порядок выполнения лабораторной работы 10
1.4. Содержание отчета по лабораторной работе 15
2. Организация пакетных файлов и сценариев в ОС Windows XP 16
2.1. Краткие теоретические сведения 16
2.2. Подготовка к выполнению лабораторной работы 20
2.3. Порядок выполнения лабораторной работы 21
2.4. Содержание отчета по лабораторной работе 28
2.5. Варианты заданий к лабораторной работе 29
3. Настройка сетевого интерфейса. Основные сетевые команды. Работа с протоколом TCP/IP в ОС Windows XP 32
3.1. Краткие теоретические сведения 32
3.2. Порядок выполнения лабораторной работы 35
3.3. Содержание отчета по лабораторной работе 37
4. Работа с ОС UNIX 39
Часть 1. Простейшие утилиты оболочки UNIX 39
4.1. Краткие теоретические сведения 39
4.1.1. Утилиты для работы с файловой структурой системы 40
4.1.2. Утилиты для работы с текстовой информацией 45
4.1.3. Утилиты для работы с файлами произвольного типа 53
4.1.4. Текстовый редактор *sed* 56
4.1.4.1. Команда вывода номера текущей строки: «=». 57
4.1.4.2. Команда добавления текста, вводимого с клавиатуры, после заданной строки: 58
4.1.4.3. Команда добавления текста, вводимого с клавиатуры, перед заданной строкой: 61
4.1.4.4. Замена строки или группы строк заданным текстом 61
4.1.4.5. Удаление заданных строк 62
4.1.4.6. Замена некоторой последовательности символов 1 на требуемую последовательность 2 62
4.1.4.7.Запись редактируемых строк в файл 64
4.2. Подготовка к выполнению части 1 лабораторной работы 66
4.3. Задание на часть 1 лабораторной работы 67
Часть 2. Базовые регулярные выражения UNIX 69
4.4. Краткие теоретические сведения 69

4.5. Структура файлов query 1 – query 5 72
4.6. Подготовка к выполнению части 2 лабораторной работы 73
4.7. Задание на часть 2 лабораторной работы 73
4.8. Содержание отчета по лабораторной работе 73
ЗАКЛЮЧЕНИЕ 75
ЛИТЕРАТУРА 76

ВВЕДЕНИЕ

Данные методические указания являются дополнением к учебному пособию "Операционные системы ЭВМ" для студентов очной формы обучения и к учебно-методическому пособию к практическим занятиям. Они являются необходимыми для практической подготовки современного инженера к деятельности по администрированию операционных систем.

Целью методических указаний является познакомить студентов с основными моментами в управлении операционными системами и привитие им соответствующих практических знаний в сфере компьютерных технологий. В современном мире без достаточно полноценного обучения работе с информацией, с принципами функционирования компьютерного оборудования не возможна качественная разработка каких-либо серьезных технических проектов.

1. Работа с файлами и дисками в ОС Windows XP

Цель работы: Изучить возможности командной оболочки и способы применения основных команд и утилит ОС Windows XP при работе с файлами и дисками.

1.1. Краткие теоретические сведения

Командная оболочка – это отдельный программный продукт, который обеспечивает прямую связь между пользователем и операционной системой (ОС). Текстовый пользовательский интерфейс в виде командной строки предоставляет среду, в которой выполняются команды, программы и служебные утилиты с текстовым интерфейсом. В командной оболочке и результат выполнения утилит и программ отображается на экране в виде, сходном с командным интерпретатором **Command.com** MS-DOS. Командная оболочка ОС Windows XP использует интерпретатор команд **Cmd.exe**, который осуществляет перевод введенной команды в понятный ОС вид, загружает приложения (утилиты) и управляет потоками данных между ними.

Имеется возможность использовать командную оболочку для создания и редактирования пакетных файлов (также называемых сценариями), что позволяет автоматизировать выполнение обычных задач. Например, можно использовать сценарии для автоматизации управления учетными записями пользователей и ежедневной архивацией в нерабочие часы. Также можно использовать сервер сценариев ОС Windows XP, **Cscript.exe**, для выполнения сложных сценариев посредством командной оболочки. Выполнение операций с помощью пакетных файлов является более эффективным, чем с помощью текстового интерфейса пользователя. Командные или пакетные файлы принимают все команды, доступные из командной строки. Дополнительные сведения о создании пакетных файлов и сценариев будут рассмотрены в соответствующей лабораторной работе.

Возможность, ориентированная непосредственно на пользователя, позволяет настроить окно командной строки для облегчения визуализации и просмотра, а также для усиления контроля текущего выполнения приложений. Чтобы реализовать эту возможность, необходимо выполнить следующие действия:

1. Загрузите командную оболочку:

– нажмите **Пуск | Выполнить**,

– наберите в появившемся окне **Cmd.exe** (или просто **cmd**),

– нажмите **Enter** для ввода.

2. Кликните правой кнопкой манипулятора «Мышь» в верхней части появившегося командного окна и выберите команду **Свойства** из контекстного меню командной оболочки.

3. В диалоговом окне **Свойства** выберите вкладку **Общие**.

4. В области **Запоминание команд** вкладки **Общие** выберите или введите значение **999** в поле **Размер буфера**, а затем выберите или введите значение **5** в поле **Количество буферов**.

5. В области **Редактирование** установите флажки **Выделение мышью** и **Быстрая вставка**.

6. В диалоговом окне **Свойства** выберите вкладку **Расположение**.

7. В области **Размер буфера экрана** вкладки **Расположение** введите или выберите значение **2500** в поле **Высота**.

8. На вкладке **Расположение** выполните следующие действия:

– в области **Размер буфера экрана** увеличьте значение параметра **Ширина**,

– в области **Размер окна** увеличьте значение параметра **Высота**,

– в области **Размер окна** увеличьте значение параметра **Ширина**,

– снимите флажок **Автоматический выбор**, а затем в области **Положение окна** измените значения полей **Левый** и **Верхний край**,

9. В диалоговом окне **Свойства** выберите вкладку **Шрифт**.

10. На вкладке **Шрифт** выполните следующие действия:

– в области **Шрифт** выберите необходимый шрифт,

– в области **Размер** выберите необходимый размер шрифта.

11. В диалоговом окне **Свойства** выберите вкладку **Цвета**.

12. На вкладке **Цвета** выполните следующие действия:

– установите флажок **Текст на экране** и выберите цвет текста, кликнув манипулятором по соответствующему полю,

– установите флажок **Фон текста** и выберите цвет фона, кликнув манипулятором по соответствующему полю,

13. Обратите внимание на то, как влияют параметры пунктов 8-12 на внешний вид командной оболочки.

14. Кликните **ОК** для ввода.

15. В диалоговом окне **Изменение свойств** выберите пункт **"Сохранить свойства для других окон с тем же именем"** или альтернативный вариант **"Изменить ярлык для запуска этого окна"** и подтвердите ввод.

При изучении возможностей командной оболочки очень важным является изучение синтаксической структуры ввода команд. Необходимо помнить, что синтаксическая структура отображается в том порядке, в котором следует вводить соответствующую команду и следующие за ней параметры, если таковые имеются.

Следующий пример команды **Xcopy** иллюстрирует разнообразие синтаксических форматов текста, а в табл. 1.1 приведены интерпретации этих форматов.

Xcopy *источник* [*результат*] [**/w**] [**/p**] [**/c**] [**/v**] [**/q**] [**/f**] [**/l**] [**/g**] [**/d**[**:***мм-дд-гггг*]] [**/u**] [**/i**] [**/s** [**/e**]] [**/t**] [**/k**] [**/r**] [**/h**] [{**/a**|**/m**}] [**/n**] [**/o**] [**/x**] [**/exclude:***файл1*[+[*файл2*]][+[*файл3*]] [{**/y**|**/-y**}] [**/z**].

Таблица 1.1. Интерпретация текстовых форматов при вводе команд

Формат	Значение
Курсив	Данные, которые должен ввести пользователь
Полужирный шрифт	Элементы, которые следует вводить точно, как показано
Пропуск (...)	Параметры могут повторяться несколько раз в командной строке
В квадратных скобках ([ ])	Необязательные элементы
В фигурных скобках ({ }); вари-анты разделены вертикальной чертой (|). Пример: {четные | нечетные}	Набор значений, из которого можно выбрать только одно значение
Шрифт Courier	Текст кода или выхода программы

Кроме того, имеется возможность вкладывать командные оболочки в **Cmd.exe**, открывая новый экземпляр **Cmd.exe** из командной строки. По умолчанию каждый экземпляр **Cmd.exe** наследует среду своего родительского приложения

Cmd.exe. Вложение экземпляров **Cmd.exe** позволяет вносить в локальную среду изменения, которые не повлияют на родительское приложение **Cmd.exe**. Это позволяет сохранять исходную среду **Cmd.exe** и возвращаться к ней после удаления вложенной командной оболочки. Изменения вложенной командной оболочки не сохраняются.

При работе с командной строкой команды являются зарезервированными словами, что означает, что нельзя объявлять переменные, имена которых совпадают с именами этих команд. Большинство команд ОС Windows XP было заимствовано разработчиками из дисковой ОС MS-DOS, которая изначально являлась операционной системой с интерфейсом командной строки и использовалась ранее на персональных компьютерах. Как и в других ОС, например в OS/2, MS-DOS позволяла преобразовывать ввод с клавиатуры в команды, организовывать такие действия, как запись и чтение с дисков, вывод на экран, управление с помощью клавиатуры и множество других внутренних операций, обеспечивающих выполнение программ и организацию файлов. В 32-битной ОС Windows XP в виде командной оболочки методом эмуляции реализован режим MS-DOS, позволяющий выполнять все указанные выше действия по работе с файлами и дисками. Кроме того, ОС Windows XP поддерживает и расширяет практически все функциональные возможности системы MS-DOS, о которых достаточно полно описано в разделе **"Новые способы выполнения типичных задач"** справки операционной системы. Дополнительную информацию по возможностям командной оболочки, а также все множество команд, доступных при работе с ней, наряду с параметрами и примерами применения можно получить в справке ОС Windows XP (**Пуск | Справка и поддержка**) в разделах **"Общие сведения о командной оболочке"**, **"Справочник по параметрам командной строки"** и **"Новые средства командной строки"**.

1.2. Подготовка к выполнению лабораторной работы

К числу основных команд и служебных утилит, используемых при работе с файлами, дисками и томами в ОС Windows XP посредством командной оболочки, относятся: **Assoc, Attrib, Cacls, Cd, Chdir, Chkdsk, Chkntfs, Comp, Compact, Convert, Copy, Date, Del, Dir, Diskcomp, Diskcopy, Erase, Fc, Find, Findstr, Format, Label, Md, Mkdir, Move, Print, Rd, Recover, Ren, Rename, Replace, Rmdir, Subst, Tree, Type, Vol, Xcopy** и другие. Дополнительная информация по этим командам, а также примеры их использования доступны в справке ОС Windows XP в соответствующих разделах.

Справку также можно получить, набрав в окне командной оболочки строку **Help** и нажав **Enter** для ввода. Полный список команд ОС Windows XP, в том числе официально не декларированных в справке ОС (например, команда **Shutdown**), может быть найден на официальном сайте корпорации Микрософт по адресу http://www.microsoft.com или непосредственно в глобальной сети Интернет.

Задачей данной лабораторной работы является демонстрация работы и ознакомление с возможностями команд, не приводящих к существенным изменениям данных на жестком диске. В настоящей лабораторной работе предполагается ознакомление с основным набором команд и служебных утилит для работы с файлами и дисками и выполнение нескольких учебных заданий с применением командной оболочки.

Перед началом выполнения лабораторной работы в среде ОС Windows XP необходимо выполнить следующее:

1) загрузить ОС Windows XP и активировать справочное меню (**Пуск | Справка и поддержка**);

2) ознакомиться с описанием и синтаксисом ввода командного интерпретатора **Cmd.exe**;

3) ознакомиться с описанием и синтаксисом ввода приведенных команд и служебных утилит.

1.3. Порядок выполнения лабораторной работы

Лабораторная работа выполняется последовательно в соответствии с определенным порядком и включает в себя одиннадцать учебных заданий.

Порядок выполнения:

I. Загрузить командную оболочку:

- нажмите **Пуск | Выполнить**,
- наберите в появившемся окне **Cmd.exe** (или просто **cmd**),
- нажмите **Enter** для ввода.

II. Изучить описание и синтаксис нижеуказанных команд в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе, либо набрав в окне командной оболочки строку ***имя_команды* /?** и нажав **Enter** для ввода.

- copy;
- xcopy;
- move;
- replace;
- ren (rename);
- fc;
- del (delete);
- erase;
- dir;
- cd (chdir);
- md (mkdir);
- rd (rmdir).

III. Выполнить следующие задания.

Задание №1.1. Исследовать основные способы применения команды копирования **Copy** на конкретных примерах.

1. Скопируйте все файлы с определенным расширением, расположенные в месте, путь к которому задайте самостоятельно, в точку назначения, заданную путем c:\Temp\.

2. Скопируйте файл, расположенный в месте, путь к которому задайте самостоятельно, в точку назначения, заданную другим путем. Инициируйте запрос на подтверждение перезаписи конечного файла в случае, если он существует.

3. Продублируйте файл с определенным именем, путь к которому задайте самостоятельно, в точку назначения, заданную тем же путем, добавив к началу имени файла строку "copy-".

4. Объедините два текстовых (.txt) файла, пути к которым задайте самостоятельно, в один файл с полным именем c:\Temp\Merged.txt.

5. Введите фрагмент текста с клавиатуры, используя ее источник **Con**, в текстовый файл, путь к которому задайте самостоятельно. Признаком конца ввода строки является **Enter**. Признаком конца ввода текста в файл являются нажатые клавиши **Ctrl+Z** и **Enter**.

6. Добавьте несколько строк с клавиатуры в конец существующего текстового файла, полученного в предыдущем пункте текущего задания.

Задание №1.2. Исследовать основные способы применения команды копирования **Xcopy** на конкретных примерах.

1. Скопируйте все файлы и подкаталоги, включая пустые и скрытые, расположенные в месте, путь к которому задайте самостоятельно, в точку назначения на другом локальном диске. При этом инициируйте запрос на подтверждение перезаписи.

2. Скопируйте дерево каталогов, включая пустые (без копирования файлов), расположенные в месте, путь к которому задайте самостоятельно, в точку назначения на другом локальном диске.

3. Скопируйте все файлы с атрибутами "архивный" и "только для чтения" с сохранением этого атрибута для файлов-результатов, расположенные в месте, путь к которому задайте самостоятельно, в точку назначения, заданную путем c:\Temp\.

4. Скопируйте все файлы и подкаталоги с датой не позднее определенной. Путь к источнику и точке назначения задайте самостоятельно. Отобразите список файлов в процессе копирования.

Задание №1.3. Исследовать основные способы применения команды перемещения **Move** на конкретных примерах.

1. Скопируйте пять любых файлов с определенным расширением, расположенные в месте источника, путь к которому выберите самостоятельно, в точку назначения, заданную путем c:\Temp\. При копировании воспользуйтесь любым методом, изученным ранее.

2. Воспользовавшись командой единожды, переместите все только что скопированные файлы, заданные путем c:\Temp\, обратно в место источника. При этом инициируйте вывод запроса на подтверждение перезаписи.

Задание №1.4. Исследовать основные способы применения команды замены **Replace** на конкретных примерах.

1. Скопируйте три любых файла, расположенные в месте каталога - источника, путь к которому выберите самостоятельно, в каждый из двух каталогов - назначения, заданных следующими путями c:\Temp\Begin\ и c:\Temp\End\. При копировании воспользуйтесь любым методом, изученным ранее.

2. Замените первый по порядку файл в каталоге - назначения c:\Temp\End\ файлом, расположенным в каталоге - источнике c:\Temp\Begin\, осуществив подтверждение замены.

3. Замените второй по порядку файл с более ранней датой модификации и путем - назначения c:\Temp\End\ файлом, расположенным в каталоге - источнике c:\Temp\Begin\, предварительно каким-либо образом его модифицировав.

4. Активируйте атрибут "только для чтения" у третьего по порядку файла в каталогах c:\Temp\Begin\ и c:\Temp\End\. Замените третий по порядку файл в каталоге - назначения c:\Temp\End\ файлом, расположенным в каталоге - источнике c:\Temp\Begin\.

Задание №1.5. Исследовать основные способы применения команды переименования **Ren** (**Rename**) на конкретных примерах.

1. Скопируйте пять любых файлов с определенными разрешениями, расположенные в месте, путь к которому выберите самостоятельно, в точку назначения, заданную путем c:\Temp\. При копировании воспользуйтесь любым методом, изученным ранее.

2. Измените типы всех скопированных файлов, заданных путем c:\Temp\, на другой, выбранный самостоятельно тип.

3. Переименуйте все файлы, заданные путем c:\Temp\, в файлы с именами Renamed1.Ren, Renamed2.Ren, … , Renamed5.Ren

Задание №1.6. Исследовать основные способы применения команды сравнения **Fc** на конкретных примерах.

1. Сравните два текстовых файла, пути к которым задайте самостоятельно. Результат сравнения выведите в файл Result.txt (используя оператор >).

2. Сравните два бинарных файла, пути к которым задайте самостоятельно. Результат сравнения добавьте в файл Result.txt (используя оператор >>).

Задание №1.7. Исследовать основные способы применения команд удаления **Del** и **Erase** на конкретных примерах.

1. Скопируйте все файлы, расположенные в месте, путь к которому выберите самостоятельно, в точку назначения, заданную путем c:\Temp\. При копировании воспользуйтесь любым методом, изученным ранее.

2. Удалите выбранный самостоятельно файл, заданный путем c:\Temp\, запросив подтверждение на удаление.

3. Удалите все файлы с атрибутом "Системный", расположенные в месте, заданном путем c:\Temp\. Подтверждение на удаление не выводить.

4. Удалите все файлы с определенным расширением, расположенные в месте, заданном путем c:\Temp\, запросив подтверждение на удаление.

5. Удалите все оставшиеся файлы, расположенные в месте, заданном путем c:\Temp\. Подтверждение на удаление не выводить.

Задание №1.8. Исследовать основные способы применения команды **Dir** на конкретных примерах.

1. Выведите постранично содержимое каталога C:\Windows\, включая вложенные подкаталоги и файлы.

2. Выведите постранично содержимое каталога C:\Windows\ в алфавитном порядке с сортировкой по столбцам и паузой после заполнения каждого экрана.

3. Выведите все файлы с расширением **.txt** каталога C:\Temp в алфавитном порядке с сортировкой по колонкам. Вывод осуществите в файл **TxtFiles.txt** (с использованием оператора >).

4. Выведите все каталоги на локальном диске C: в алфавитном порядке. Результат добавьте в файл **TxtFiles.txt** (с использованием оператора >>).

5. Добавьте сведения о владельцах файлов системного каталога C:\Windows\ в файл DocFiles.txt (с использованием оператора >>).

Задание №1.9. Исследовать основные способы применения команды перехода в другой каталог **Cd** (**ChDir**) на конкретных примерах.

1. Смените текущий каталог на каталог, полный путь к которому задан следующим образом C:\WINDOWS\Help\Tours\WindowsMediaPlayer\Video\.

2. Перейдите из подкаталога ..\Video на уровень выше.

3. Смените текущий каталог на каталог, полный путь к которому задан следующим образом C:\WINDOWS\Help\Tours\WindowsMediaPlayer\Audio\.

4. Перейдите из подкаталога …\Audio на два уровня выше одной командой с использованием относительных ссылок.

5. Смените текущий локальный диск на диск D:

Задание №1.10. Исследовать основные способы применения команды создания каталога **Md** (**MkDir**) на конкретных примерах.

1. Создайте каталог, путь к которому выберите самостоятельно.

2. Единожды воспользовавшись командой, создайте каталог, полный путь к которому задан следующим образом c:\Temp*номер_группы*\MyPath\.

Задание №1.11. Исследовать основные способы применения команды удаления каталога **Rd** (**RmDir**) на конкретных примерах.

1. Удалите подкаталог третьего уровня **MyPath**, созданный в предыдущем задании №1.10.

2. Скопируйте несколько файлов, расположенных в месте, путь к которому выберите самостоятельно, в точку назначения, заданную путем c:\Temp\VMGroup\. При копировании воспользуйтесь любым методом, изученным ранее.

3. Единожды воспользовавшись командой, без запроса подтверждения удалите дерево каталогов c:\Temp\VMGroup\, включая подкаталог второго уровня VMGroup с содержащимися внутри файлами.

Внимание!!! При выполнении каждого из заданий **1.1 – 1.11** используйте следующие инструкции:

– по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,

– нажмите **Enter** для ввода,

– изучите полученный результат и сделайте вывод о проделанной работе,

– запишите полученную информацию в отчет, заполнив табл. 1.2.

Таблица 1.2. Результаты выполнения команд

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		
6.		

1.4. Содержание отчета по лабораторной работе

Отчет по лабораторной работе оформляется в соответствии с требованиями государственного стандарта и должен содержать:

1) титульный лист;

2) описание и цель работы;

3) краткое описание служебных команд и утилит, предназначенных для работы с файлами и дисками в среде командной оболочки;

4) результаты исследований работы служебных команд и утилит в соответствии с учебными заданиями лабораторной работы;

5) заполненные таблицы учебных заданий лабораторной работы;

6) выводы о проделанной работе.

2. Организация пакетных файлов и сценариев в ОС Windows XP

Цель работы: Изучить принципы построения и организации пакетных файлов и сценариев в среде ОС Windows XP.

2.1. Краткие теоретические сведения

Пакетный файл это неформатированный текстовый файл ASCII, содержащий одну или несколько команд ОС. Имена пакетных файлов имеют расширения .cmd или .bat. ОС при работе с пакетным файлом последовательно обрабатывает его команды после ввода его имени в строке командной оболочки или запуска из другой программы.

Другой разновидностью пакетного файла является сценарий, представляющий собой программу, состоящую из набора инструкций для работы приложения или служебной утилиты. Инструкции в сценариях обычно выражаются с использованием правил и синтаксиса соответствующего приложения или служебной утилиты в сочетании с простыми управляющими операторами, такими как операторы циклов и условные операторы.

Пакетные файлы и сценарии часто называют командными файлами, содержащими любые команды. Некоторые команды, такие как **For**, **Goto** и **If**, позволяют выполнять обработку условий в пакетных файлах. В частности, **If** позволяет запускать команды в зависимости от выполнения заданного условия. Другие команды позволяют управлять вводом и выводом, а также запускать другие пакетные файлы. Совместно с командами, изученными в предыдущей лабораторной работе, вышеуказанные команды позволяют создавать пакетные файлы практически для любых целей управления работой и администрирования ОС Windows XP.

Следующее, что необходимо отметить при организации пакетных файлов и сценариев, является применение переменных, задающих поведение командной оболочки или ОС в целом и так называемых пакетных параметров командного интерпретатора, которые используются в пакетном файле для получения информации о настройках среды.

Имеется возможность определить поведение среды командной оболочки или всей ОС с помощью двух типов переменных среды: *системных и локальных*. Системные переменные определяют поведение глобальной среды ОС. Локальные

переменные определяют поведение среды в конкретном экземпляре командного интерпретатора **Cmd.exe**.

Системные переменные среды задаются заранее в ОС Windows XP и доступны для всех ее процессов. Только пользователи с привилегиями администратора могут изменять эти переменные.

Локальные переменные среды доступны в случае, когда пользователь, для которого они были созданы, входит в систему. В частности, локальные переменные реестра **HKEY_CURRENT_USER** подходят только для текущего пользователя, но определяют поведение глобальной среды ОС.

В следующем списке представлены различные типы переменных в порядке убывания их приоритета:

- встроенные системные переменные,
- системные переменные реестра **HKEY_LOCAL_MACHINE**,
- локальные переменные реестра **HKEY_CURRENT_USER**,
- все переменные среды и пути, указанные в файле Autoexec.bat,
- все переменные среды и пути, указанные в сценарии входа в систему, если он имеется,
- переменные, используемые интерактивно в пакетном файле или сценарии.

Чтобы иметь возможность подставить значение в переменную среды из командной строки или в пакетном файле (сценарии), следует заключить имя соответствующей переменной (**Приложение 1**) в символы процентов (%), например **Set** MyPath=%CD%. Символы процентов указывают на то, что командный интерпретатор должен обратиться к значению переменной без посимвольного ее разложения и сравнения.

Командный интерпретатор **Cmd.exe** может оперировать переменными с %0 по %9. При использовании пакетных параметров переменная %0 заменяется именем пакетного файла, а переменные с %1 по %9 – на соответствующие аргументы командной строки. Для доступа к переменным больше %9 необходимо воспользоваться командой **Shift**. Параметр %* ссылается на все аргументы, которые передаются пакетному файлу, за исключением %0.

В качестве примера, рассмотрим копирование содержимого из каталога 1 (Folder1) в каталог 2 (Folder2), где параметр %1 заменяется значением Folder1, а параметр %2 соответственно значением Folder2. В пакетном файле **Mybatch.bat** следует ввести следующую строку:

Xcopy %1*.* %2

Используйте пакетный файл Mybatch.bat следующим образом:

Mybatch.bat C:\folder1 D:\folder2

Результат будет таким же, как и при записи в пакетный файл строки:

Xcopy C:\folder1*.* D:\folder2\

С пакетными параметрами можно также использовать модификаторы. Модификаторы используют информацию о текущем диске и каталоге как часть или полное имя файла (каталога).

Синтаксис модификатора: **%~xy,** где **x –** символьное сокращение действия, определяемое модификатором, **y** – идентификатор переменной (в диапазоне от 1 до 9).

В табл. 2.1 и 2.2 описаны модификаторы, выполняемые ими действия, и даны возможные комбинации модификаторов и квалификаторов для получения более сложных результатов. В этих таблицах %1 и переменную среды PATH можно заменить другими значениями пакетных параметров.

Таблица 2.1. Модификаторы и выполняемые ими действия

№ п/п.	Модификатор	Описание
1.	%~1	расширение %1 и удаление любых кавычек (" ")
2.	%~f1	замена %1 полным путем
3.	%~d1	замена %1 именем диска
4.	%~p1	замена %1 путем
5.	%~n1	замена %1 именем файла
6.	%~x1	замена %1 расширением имени файла
7.	%~s1	замена путем, содержащим только короткие имена
8.	%~a1	Замена %1 атрибутами файла
9.	%~t1	замена %1 датой и временем модификации файла
10.	%~z1	замена %1 размером файла
11.	%~$PATH:1	поиск в каталогах, перечисленных в переменной среды PATH, замена %1 полным именем первого найденного файла. Если переменная среды не определена или поиск не обнаружил файлов, модификатор выдает пустую строку.

Таблица 2.2. Комбинации модификаторов и квалификаторов

№ п/п.	Модификатор	Описание
1.	%~dp1	замена %1 именем диска и путем
2.	%~nx1	замена %1 именем файла и расширением
3.	%~dp$PATH:1	поиск в каталогах, перечисленных в переменной среды PATH, и замена %1 именем диска и путем к первому найденному файлу.
4.	%~ftza1	замена %1 строкой, аналогичной результату работы команды **Dir**

Еще один модификатор, являющийся уникальным, имеет вид %*. Он представляет все аргументы, переданные пакетному файлу. Этот модификатор не используется в комбинации с модификатором %~.

Подводя промежуточные итоги по теоретическому материалу, необходимо напомнить еще о двух возможностях, а именно о конвейерах команд и "каналах", рассмотренных в предыдущей лабораторной работе. Наряду с рассмотренными командами и утилитами, модификаторами и квалификаторами, они являются инструментами для расширения функционала пакетных файлов и сценариев при их построении и организации.

Исполняющим механизмом, позволяющим реализовать задуманные в пакетном файле или сценарии действия, является сервер сценариев ОС Windows XP, который позволяет быстро запустить пакетный файл или сценарий, введя его имя в строке командной оболочки. Сервер сценариев служит контроллером средств обработки сценариев в ОС Windows XP. Однако, в отличие от других средств обработки сценариев, сервер сценариев ОС Windows XP не требует много памяти и является идеальным средством, как для интерактивных, так и для пакетных сценариев, таких как сценарий входа в систему или сценарий администрирования.

Существуют две версии сервера сценариев, доступных в окне командной оболочки: ***Wscript.exe*** – позволяет задавать параметры выполнения сценариев в окне свойств, и ***Cscript.exe*** – позволяет задавать параметры выполнения сценариев с помощью ключей командной строки.

В ранних версиях ОС Windows XP в качестве языка сценариев поддерживался только язык команд MS-DOS. По сравнению с новыми языками VBScript и JScript, язык команд MS-DOS обладает ограниченным набором средств, хотя и является более компактным и быстрым. В частности, в MS-DOS нет средств для управления процессом выполнения программы, в то время как сервер сценариев ОС Windows XP,

основанный на мощном языке VBScript (JScript), позволяет воспользоваться подобными преимуществами и при этом поддержка языка команд MS-DOS по-прежнему остается доступной.

Для разработки сценариев ОС Windows XP следует использовать редакторы сценариев JScript или VBScript (в составе Visual Basic Scripting Edition). При запуске сценария из командной строки, сервер сценария читает и передает содержимое указанного файла зарегистрированному обработчику сценариев. Для определения языка сценария используется расширение имени файла (.vbs для VBScript, .js для JScript). Благодаря этому, разработчик сценария не обязан знать точные программные идентификаторы (ProgID) различных обработчиков сценариев. Сопоставление расширения имени файла сценария с программным идентификатором и запуск конкретного обработчика сценариев осуществляется непосредственно сервером сценариев ОС Windows XP.

В рамках настоящей лабораторной работы не предполагается использование среды Visual Basic для написания сценариев, поскольку, с одной стороны, эта среда изучается в рамках отдельного курса, а с другой стороны, зная общие принципы построения и организации пакетных файлов (сценариев) и имея достаточно обширную базу примеров, доступных на сайте Microsoft, без труда можно исследовать их работу и, в случае необходимости, внести в необходимый скрипт изменения, отражающие специфику поставленной задачи.

Простейшим сценарием, не требующим применения среды Visual Basic, является сценарий входа в систему, представляющий собой файл, связываемый с одной или несколькими учетными записями пользователей. Обычно сценарий входа является пакетным файлом, который автоматически выполняется при каждом входе пользователя в систему. Сценарии входа используются для настройки рабочей среды пользователя при входе и позволяют администратору задавать основные параметры рабочей среды пользователя без непосредственного его участия.

2.2. Подготовка к выполнению лабораторной работы

Поскольку пакетные файлы могут включать в себя любые команды, их конвейеры и "каналы", при большом количестве условий и циклов последствия некорректной работы пакетного файла могут быть непредсказуемыми для ОС, и, возможно, как следствие – разрушительными. Поэтому для организации пакетного файла разработчику необходимо четко представлять себе, что именно и каким

образом должно происходить в системе при работе этого файла, какая последовательность действий реализуется в результате выполнения задуманного сценария и как на эти действия реагирует ОС.

Помимо рассмотренных в предыдущей лабораторной работе команд, которые могут быть использованы при организации пакетного файла, существует ряд дополнительных, функционал которых напоминает операторы языков программирования высокого уровня. К их числу относятся: **At, Call, Doskey, Echo, Endlocal, For, Goto, If, Pause, Rem, Set, Setlocal** и **Shift.**

В настоящей лабораторной работе предполагается ознакомление с основными командами, используемыми в качестве инструментов организации пакетных файлов, создание командного файла в формате ASCII, реализующего определенный сценарий работы системы, а также оценка возможности использования его в качестве сценария входа в систему.

Перед началом выполнения лабораторной работы в среде ОС Windows XP необходимо выполнить следующее:

1) загрузить ОС Windows XP и активировать справочное меню **(Пуск | Справка и поддержка)**;

2) ознакомиться с описанием и синтаксисом ввода командного интерпретатора **Cmd.exe**;

3) ознакомиться с описанием и синтаксисом ввода приведенных команд и служебных утилит.

2.3. Порядок выполнения лабораторной работы

Лабораторная работа выполняется последовательно в соответствии с определенным порядком и включает в себя восемь учебных задания.

Порядок выполнения:

I. Загрузить командную оболочку:

- нажмите **Пуск | Выполнить**,
- наберите в появившемся окне **Cmd.exe** (или просто **cmd**),
- нажмите **Enter** для ввода.

II. Изучить описание и синтаксис нижеуказанных команд в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе, либо

набрав в окне командной оболочки строку ***имя_команды*** **/?** и нажав **Enter** для ввода.

- set;
- rem;
- echo;
- for;
- if;
- goto;
- call;
- setlocal;
- endlocal.

III. Выполнить следующие задания.

Задание №2.1. Исследовать способы применения команды присвоения переменной среды **Set** на конкретных примерах.

1. Отобразите переменные среды двумя способами: из командной оболочки и окна свойств системы (**Пуск | Панель управления | Система)**.

2. Задайте переменную среды, содержащую определенный путь к месту назначения, выбранный самостоятельно.

3. Проверьте наличие в системе переменной среды, заданной в предыдущем пункте задания.

4. Выведите значение выражения, определенного в соответствии с вариантом задания (**подраздел 2.5**), в качестве переменной среды **Result**.

5. Задайте переменную среды с различными вариантами динамически формируемых значений (табл. 2.3). Варианты динамических значений выберите самостоятельно.

Таблица 2.3. Динамические значения команды **Set**

Значение	Описание действия
%Cd%	раскрывается в строку текущей директории
%Date%	раскрывается в текущую дату
%Time%	раскрывается в текущее время
%Random%	раскрывается в случайное десятичное число в диапазоне от 0 до 32767
%Errorlevel%	раскрывается в текущее значение ErrorLevel
%Cmdextversion%	раскрывается в текущее значение версии расширенной обработки команд
%Cmdcmdline%	раскрывается в исходную командную строку, которая вызвала текущее окно командной оболочки

При выполнении задания используйте следующие инструкции:

– по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,

– нажмите **Enter** для ввода,

– изучите полученный результат и сделайте вывод о проделанной работе,

– запишите полученную информацию в отчет, заполнив таблицу 2.4.

Задание №2.2. Исследовать способы применения команды отображения текста **Echo** на конкретных примерах.

1. Создайте пакетный файл, воспользовавшись любым текстовым редактором. Имя пакетного файла выберете самостоятельно.

2. Введите в созданный пакетный файл текст, приведенный ниже.

Cls

@Echo off

Echo.

Rem ****** Эта пакетная программа ******

Rem ***** иллюстрирует возможности *****

Rem ********** команды Echo **********

Echo.

Echo ******** This batch program ********

Echo *** illustrates possibilities of ***

Echo ********* the Echo command *********

Echo.

Pause

3. Сохраните текст пакетного файла.

При выполнении задания используйте следующие инструкции:

– воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,

– изучите пример и полученный с его помощью результат, обратив внимание на то, что команда **Echo** с точкой (.) в конце выводит на экран пустую строку, а символ "коммерческое И" (@) перед командой **Echo** отключает режим отображения команд.

– сделайте вывод о проделанной работе и запишите его в отчет.

Задание №2.3. Исследовать способы применения команды циклической обработки данных **For** на конкретных примерах.

1. Скопируйте файлы каталога, путь к которому задайте самостоятельно, в точку назначения, заданную путем C:\Documents and Settings\student\Temp\. При копировании воспользуйтесь любым методом, изученным ранее.

2. К каждому из файлов, местоположение которых определено путем C:\Documents and Settings\student\Temp\, добавьте символ «!» в начале имени, воспользовавшись командой циклической обработки данных.

3. Подсчитать количество каталогов на локальном диске, воспользовавшись командой циклической обработки данных, в процессе выполнения выводя результат в переменную среды, выбранную самостоятельно. Проверьте полученный результат в файловом диспетчере **Проводник** (**Правая кнопка мыши | Свойства**).

4. Модифицируйте пакетный файл, полученный в предыдущем задании, воспользовавшись командой циклической обработки данных таким образом, чтобы в процессе его выполнения отображалось определенное количество раз выражение «********* the For command **********».

При выполнении задания используйте следующие инструкции:

– по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,

– нажмите **Enter** для ввода,

– изучите полученный результат и сделайте вывод о проделанной работе,

– запишите полученную информацию в отчет, заполнив таблицу 2.4.

Задание №2.4. Исследовать способы применения команды обработки условия **If** на конкретных примерах.

Модифицируйте пакетный файл, полученный в предыдущем задании таким образом, чтобы выполнялись следующие условия:

1. Если не существует каталог C:\Documents and Settings\student\Temp\MyFont\, создайте его любым способом, изученным ранее. В противном случае выведите сообщение "Folder exists" (Каталог существует).

2. Если в каталоге C:\Documents and Settings\student\Temp\MyFont\ не существует файлов-шрифтов, скопируйте любые три одним из методов, изученных ранее, из системного каталога c:\Windows\Fonts\. В противном случае выведите сообщение "Fonts exist" (Шрифты присутствуют).

3. Если в каталоге C:\Documents and Settings\student\Temp\MyFont\ существует файлы, удалите каталог вместе с его содержимым, изученным ранее способом и выведите сообщение "Folder deleted". В противном случае выведите сообщение "Folder is empty. Deleting is senseless» (Каталог пуст. Удаление бессмысленно).

При выполнении задания используйте следующие инструкции:

– по каждому из пунктов задания в командном файле наберите соответствующий код из команд с необходимыми ключами,

– сохраните модифицированный пакетный файл,

– воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,

– изучите полученный результат и сделайте вывод о проделанной работе,

– запишите полученную информацию в отчет, заполнив таблицу 2.4.

Задание №2.5. Исследовать способы применения команды перехода **Goto** на конкретных примерах.

1. Модифицируйте существующий пакетный файл, введя в него следующий текст:

Pause

Echo.

Format A:
If not Errorlevel 1 **Goto** End
Echo.
Echo *** Error of formatting ***
Rem *** Ошибка форматирования ***
:End
Echo.
Echo *** The end of batch program ***
Rem *** Конец пакетной программы ***
Echo.
Pause

2. Сохраните текст пакетного файла.

При выполнении задания используйте следующие инструкции:

– воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,
– изучите пример и полученный с его помощью результат,
– сделайте вывод о проделанной работе и запишите его в отчет.

Задание №2.6. Исследовать способы применения команды вызова пакетного файла **Call** на конкретных примерах.

1. Создайте новый (дочерний) пакетный файл, воспользовавшись любым текстовым редактором. Имя пакетного файла выберите самостоятельно.

2. Введите в дочерний пакетный файл процедуру форматирования гибкого диска, учитывающую переход в начало процедуры в случае ошибки, из приведенного выше примера.

3. Модифицируйте родительский пакетный файл, удалив из него лишние команды и добавив ссылку на дочерний пакетный файл для его вызова.

4. Сохраните тексты обоих пакетных файлов.

При выполнении пунктов 1-4 задания используйте следующие инструкции:

– воспользовавшись командой **Start** и указав путь к родительскому файлу, запустите его на выполнение, нажав **Enter** для ввода,

– изучите полученный результат и сделайте вывод о проделанной работе,

– запишите полученную информацию в отчет.

5. Вспомните команду копирования **Xcopy** и ее параметры.

6. Модифицируйте родительский и дочерний файлы таким образом, чтобы осуществилась передача из родительского файла двух значений параметров (%переменная) команды **Xcopy**, находящейся внутри дочернего файла.

7. Сохраните тексты обоих пакетных файлов.

При выполнении пунктов 5-7 задания используйте следующие инструкции:

– воспользовавшись командой **Start** и указав путь к родительскому файлу с параметрами для команды **Xcopy**, запустите его на выполнение, нажав **Enter** для ввода,

– изучите полученный результат и сделайте вывод о проделанной работе,

– перенесите тексты модифицированных пакетных файлов, а также значения используемых пакетных параметров в отчет.

Задание №2.7. Исследовать применение команд локализации переменных среды **Setlocal** и **Endlocal** на конкретном примере.

1. Модифицируйте существующий пакетный файл, введя в него следующий текст, иллюстрирующий локальное изменение переменных среды:

@Echo off

Echo.

Echo *** Local changing the environment variables ***

Rem *** Локальное изменение переменных среды ***

Setlocal

Path=c:\Windows\system32\help;%path%

Call help>c:\help.out

Endlocal

Start notepad c:\help.out

Pause

2. Сохраните текст пакетного файла.

При выполнении задания используйте следующие инструкции:

– воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,

– изучите пример и полученный с его помощью результат,

– сделайте вывод о проделанной работе и запишите его в отчет.

Дополнительную информацию по возможностям командной оболочки, а также все множество команд доступных при работе с ней наряду с параметрами и примерами применения можно получить в справке ОС Windows XP **(Пуск | Справка и поддержка)** в разделах **«Общие сведения о командной оболочке», «Справочник по параметрам командной строки»** и **«Новые средства командной строки».**

2.4. Содержание отчета по лабораторной работе

Отчет по лабораторной работе оформляется в соответствии с требованиями государственного стандарта и должен содержать:

1) титульный лист;

2) описание и цель работы;

3) краткое описание служебных команд и утилит командной оболочки, предназначенных для построения и организации пакетных файлов и сценариев в среде ОС Windows XP;

4) результаты исследований работы служебных команд и утилит в соответствии с учебными заданиями лабораторной работы;

5) заполненные таблицы учебных заданий лабораторной работы (таблица 2.4).

Таблица 2.4. Результаты выполнения команды

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		

6) алгоритмы, блок-схемы и тексты пакетных файлов;

7) выводы о проделанной работе.

2.5. Варианты заданий к лабораторной работе

Варианты заданий для выполнения лабораторной работы представлены в табл. 2.5. Обратите внимание, что все числа в таблице – шестнадцатеричные.

Таблица 2.5. Варианты заданий к лабораторной работе

Вар-т	A	b	c	Result
1.	24	11	35	(a + b – c) * 10
2.	AA	01	C1	a * 5 – b / 5 + c
3.	12	33	10	a * 4 / c – b * 2
4.	25	A3	B4	a – b * 3 – c / 2
5.	49	02	65	a – b * b + c / 5
6.	21	99	12	(b *(a + c)) / 3
7.	BC	BC	CB	10 + (a – b) * c
8.	01	94	04	(b – c) / (a * 9)
9.	84	D2	2A	a / 10 – (b * c)
10.	10	39	92	a * a + b – c / 2
11.	D1	CC	1C	a * b * c – b / c
12.	FF	00	F1	(a – c) * b + 25
13.	CA	DA	FA	a * b * (FA – c)
14.	45	78	87	((c – b) *a) / 8
15.	88	88	00	(a – b) / (1 – c)
16.	75	93	02	(b + c) * 2 –a
17.	A1	CD	0E	a*a*b*b*c*c
18.	C4	EA	E3	(a + b + c) * a
19.	44	55	33	c *(b – a) / 10
20.	B3	E5	DE	(a * b * c) / 25
21.	BB	ED	AE	(a – b) * 2 – c
22.	55	56	31	(b – a) * 31 – c
23.	D1	EC	EE	(c – b) * 5 / a
24.	D6	E6	FE	(a + b) * c – 11
25.	71	65	32	(a – b) *c / 32
26.	84	32	10	(a * (b + c)) / 5

Приложение 1. Системные и локальные переменные среды ОС Windows XP.

Полный список системных и локальных переменных среды, применяемых в ОС Windows XP, приведен в табл. П.1.

Таблица П.1. Переменные среды ОС Windows XP

№ п/п.	Переменная	Тип	Описание
1.	%Allusersprofile%	Local	Возвращает размещение профиля «All Users».
2.	%Appdata%	Local	Возвращает размещение данных приложений.
3.	%Cd%	Local	Возвращает путь к текущей папке.
4.	%Cmdcmdline%	Local	Возвращает строку команд, с помощью которой был запущен данный экземпляр командного интерпретатора.
5.	%Cmdextversion%	System	Возвращает номер версии текущих расширений командного интерпретатора
6.	%Computername%	System	Возвращает имя компьютера.
7.	%Comspec%	System	Возвращает путь к исполняемой командной оболочке.
8.	%Date%	System	Возвращает текущую дату.
9.	%Errorlevel%	System	Возвращает код ошибки последней использовавшейся команды (≥0 – ошибка).
10.	%Homedrive%	System	Возвращает имя диска локальной рабочей станции, связанного с основным каталогом пользователя.
11.	%Homepath%	System	Возвращает полный путь к основному каталогу пользователя.
12.	%Homeshare%	System	Возвращает сетевой путь к общему основному каталогу пользователя.
13.	%Logonsever%	Local	Возвращает имя контроллера домена, проверявшего подлинность сессии

Продолжение таблицы П.1.

№ п/п.	Переменная	Тип	Описание
14.	%Number_of_processors%	System	Возвращает число установленных процессоров.
15.	%Os%	System	Возвращает имя ОС.
16.	%Path%	System	Указывает путь поиска для исполняемых файлов.
17.	%Pathext%	System	Возвращает список расширений исполняемых файлов.
18.	%Processor_architecture%	System	Возвращает архитектуру процессора.
19.	%Processor_identfier%	System	Возвращает описание процессора.
20.	%Processor_level%	System	Возвращает номер модели процессора.
21.	%Processor_revision%	System	Возвращает номер модификации процессора.
22.	%Prompt%	Local	Возвращает параметры командной строки для текущего интерпретатора.
23.	%Random%	System	Возвращает произвольное десятичное число от 0 до 32767.
24.	%Systemdrive%	System	Возвращает имя диска, содержащего системный каталог ОС.
25.	%Systemroot%	System	Возвращает размещение системного каталога ОС.
26.	%Temp% и %Tmp%	System	Возвращает доступные временные папки.
27.	%Time%	System	Возвращает текущее время.
28.	%Userdomain%	Local	Возвращает имя домена, содержащего список учетных записей пользователей.
29.	%Username%	Local	Возвращает имя пользователя в системе
30.	%Userprofile%	Local	Возвращает размещение профиля для текущего пользователя.
31.	%Windir%	System	Возвращает размещение каталога ОС.

3. Настройка сетевого интерфейса. Основные сетевые команды. Работа с протоколом TCP/IP в ОС Windows XP

Цель работы: Изучить принципы настройки и проверки функционирования локальной сети средствами ОС Windows XP.

3.1. Краткие теоретические сведения

TCP/IP (Transmission Control Protocol / Internet Protocol) является самым популярным сетевым протоколом, служащим основой глобальной сети Интернет. Предлагаемые им средства маршрутизации обеспечивают максимальную гибкость функционирования локальных сетей предприятий. В ОС Windows XP протокол **TCP/IP** устанавливается автоматически. В сетях протокола **TCP/IP** каждому клиенту должен быть назначен соответствующий **IP**-адрес, представляющий собой 32-разрядное число, разделенное точками (например, 192.168.1.255). Кроме того, клиенту может потребоваться служба имен или алгоритм разрешения имен. В комплект протокола **TCP/IP** входят служебные программы **FTP** (File Transfer Protocol) и **Telnet**. **FTP** – это приложение с текстовым интерфейсом, позволяющее подключаться к FTP-серверам и передавать файлы. **Telnet** обладает графическим интерфейсом и позволяет входить на удаленный компьютер и выполнять команды так же, как если бы пользователь находился за клавиатурой этого компьютера.

В стеке TCP/IP используются три типа адресов: **локальные** (называемые также **аппаратными**), **IP-адреса** и **символьные доменные** имена.

В терминологии TCP/IP под **локальным адресом** понимается такой тип адреса, который используется средствами базовой технологии для доставки данных в пределах подсети, являющейся элементом составной интерсети. В разных подсетях допустимы разные сетевые технологии, разные стеки протоколов, поэтому при создании стека TCP/IP предполагалось наличие разных типов локальных адресов. Если подсетью интерсети является локальная сеть, то локальный адрес – это **MAC-адрес**. MAC-адрес назначается сетевым адаптерам и сетевым интерфейсам маршрутизаторов. MAC-адреса назначаются производителями оборудования и являются уникальными, так как управляются централизованно. Для всех существующих технологий локальных сетей MAC-адрес имеет формат 6 байт, например 11-A0-17-3D-BC-01. Однако протокол IP может работать и над протоколами более высокого уровня, например над протоколом IPX или X.25. В этом

случае локальными адресами для протокола IP соответственно будут адреса IPX и X.25. Следует учесть, что компьютер в локальной сети может иметь несколько локальных адресов даже при одном сетевом адаптере. Некоторые сетевые устройства не имеют локальных адресов. Например, к таким устройствам относятся глобальные порты маршрутизаторов, предназначенные для соединений типа "точка-точка".

IP-адреса представляют собой основной тип адресов, на основании которых сетевой уровень передает пакеты между сетями. Эти адреса состоят из 4 байт, например 109.26.17.100. IP-адрес назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Internet Network Information Center, InterNIC), если сеть должна работать как составная часть Internet. Обычно поставщики услуг Internet получают диапазоны адресов у подразделений InterNIC, а затем распределяют их между своими абонентами. Номер узла в протоколе IP назначается независимо от локального адреса узла. Маршрутизатор по определению входит сразу в несколько сетей. Поэтому каждый порт маршрутизатора имеет собственный IP-адрес. Конечный узел также может входить в несколько IP-сетей. В этом случае компьютер должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом, IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

Символьные доменные имена. Символьные имена в IP-сетях называются доменными и строятся по иерархическому признаку. Составляющие полного символьного имени в IP-сетях разделяются точкой и перечисляются в следующем порядке: сначала простое имя конечного узла, затем имя группы узлов (например, имя организации), затем имя более крупной группы (поддомена) и так до имени домена самого высокого уровня (например, домена объединяющего организации по географическому принципу: RU - Россия, UK - Великобритания, US - США), Примеров доменного имени может служить имя base2.sales.zil.ru. Между доменным именем и IP-адресом узла нет никакого алгоритмического соответствия, поэтому необходимо использовать какие-то дополнительные таблицы или службы, чтобы узел сети однозначно определялся как по доменному имени, так и по IP-адресу. В сетях TCP/IP используется специальная распределенная служба Domain Name System (**DNS**),

которая устанавливает это соответствие на основании создаваемых администраторами сети таблиц соответствия. Поэтому доменные имена называют также **DNS-именами**.

Для выделения в IP-адресах адресов сети и адресов узла применяется **маска подсети**. *Маска* – это число, которое используется в паре с IP-адресом; двоичная запись маски содержит единицы в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети. Поскольку номер сети является цельной частью адреса, единицы в маске также должны представлять непрерывную последовательность.

В масках количество единиц в последовательности, определяющей границу номера сети, не обязательно должно быть кратным 8, чтобы повторять деление адреса на байты. Пусть, например, для IP-адреса 129.64.134.5 указана маска 255.255.128.0, то есть в двоичном виде:

IP-адрес 129.64.134.5 - 10000001. 01000000.10000110. 00000101

Маска 255.255.128.0 - 11111111.11111111.10000000. 00000000

То есть в десятичной форме записи - номер сети 129.64.128.0, а номер узла 0.0.6.5.

Механизм масок широко распространен в IP-маршрутизации, причем маски могут использоваться для самых разных целей. С их помощью администратор может структурировать свою сеть, не требуя от поставщика услуг дополнительных номеров сетей. На основе этого же механизма поставщики услуг могут объединять адресные пространства нескольких сетей путем введения так называемых "префиксов" с целью уменьшения объема таблиц маршрутизации и повышения за счет этого производительности маршрутизаторов.

Служебные программы и утилиты протокола **TCP/IP** обеспечивают подключение к различным современным сетям. При этом чтобы использовать эти утилиты, на компьютере должна быть установлена поддержка протокола **TCP/IP**. К числу поддерживаемых протоколом **TCP/IP** служебных команд и утилит относятся следующие: **Finger, Ping, Ftp, Rcp, Hostname, Rexec, Ipconfig, Route, Lpq, Rsh, Lpr, Tftp, Nbtstat, Tracert, Netstat, Getmac,** а также целый ряд команд с приставкой **Net [accounts | computer | config | continue | file | group | help | helpmsg | localgroup | name | pause | print | send | session | share | start | statistics | stop | time | use | user | view]** и другие**.** Дополнительные сведения о запуске служб **TCP\IP** из командной строки находятся в разделе **Net start**.

В настоящей лабораторной работе предполагается ознакомление с основным набором команд протокола **TCP/IP** и выполнение нескольких учебных заданий с применением командной оболочки.

3.2. Порядок выполнения лабораторной работы

После завершения запуска ОС необходимо выполнить следующие действия:

1. Вызвать свойства сетевого окружения (щелчок правой кнопкой "мыши" на ярлыке "Сетевое окружение" – "Свойства").
2. На закладке "Общие" выбираем **Internet Protocol (TCP/IP)** – "Свойства" и задать следующие параметры сетевого интерфейса:

IP-адрес:

если № варианта состоит из одной цифры – 192.168.220.20*№варианта*

если № варианта состоит из двух цифр – 192.168.220.2*№варианта*

Маска подсети: 255.255.255.0

Шлюз по умолчанию: 192.168.220.2

DNS-серверы: 192.168.220.10

192.168.220.11

3. Подтвердить выполненные настройки.
4. Выполнить проверку настроек. Для этого необходимо проделать следующее:
   - запустить консоль командной строки;
   - изучить синтаксис команды **ipconfig**;
   - выполнить проверку настроек, выписав локальный адрес (адрес физической машины) и IP-адрес виртуальной машины.
5. Выполнить проверку связи с другими узлами. Для этого необходимо проделать следующее:
   - запустить консоль командной строки (если после выполнения предыдущего пункта консоль была закрыта);
   - изучить синтаксис команды **ping**;
   - выполнить проверку связи с физической локальной машиной, используя ip-адреса узлов.
6. Исследовать содержимое кэша **ARP**. Для этого в окне командной оболочки выполните следующие действия:
   - наберите команду **Arp** с необходимыми ключами;

– нажмите **Enter** для ввода;

– самостоятельно осуществите добавление в кэш **ARP** статической записи со следующими параметрами:

ip-адрес 192.168.220.13,

MAC-адрес 00-04-23e2-ac-a7;

– повторно исследуйте содержимое кэша **ARP**,

7. Вывести список интерфейсов и их индексов. Для этого в окне командной оболочки выполните следующие действия:

– наберите команду **Route** с необходимыми ключами,

– нажмите **Enter** для ввода,

8. Проверить наличие соединения с узлом сети по заданному **IP**-адресу или имени узла. Для этого в окне командной оболочки выполните следующие действия:

– наберите команду **Ping** с необходимыми ключами согласно условиям:

число отправляемых сообщений с эхо-запросом – 10,

длина поля данных – 4096 байт;

– нажав **Enter** для ввода, проверьте наличие соединения с узлом сети, имеющего:

IP-адрес петли обратной связи,

IP-адрес собственного узла пользователя,

IP-адрес основного шлюза (по умолчанию),

IP-адрес сайта www.mail.ru,

9. Выполнить трассировку маршрута до определенной точки назначения, заданной **IP**-адресом или именем узла. Исследовать статистику переходов и потерь **TCP/IP**-пакетов в процессе трассировки. Для этого в окне командной оболочки выполните следующие действия:

– наберите команду **Tracert** с необходимыми ключами,

– нажав **Enter** для ввода, выполните трассировку маршрута, имеющего:

IP-адрес шлюза (маршрутизатора) внешнего сетевого интерфейса,

IP-адрес физического локального узла,

имя удаленного узла внешней сети, принадлежащего томскому сегменту сети Интернет,

– наберите команду **Pathping** с необходимыми ключами,

– нажав **Enter** для ввода, выполните трассировку маршрута, имеющего тот же **IP**-адрес или имя удаленного узла внешней сети, принадлежащего томскому сегменту сети Интернет,

10. Исследовать статистические данные **TCP/IP**-подключений с помощью команды **Netstat** на конкретных примерах:

1. Выведите **Ethernet** статистику**.**

2. Выведите статистику по всем активным протоколам.

3. Выведите статистику только по **TCP**-протоколу.

4. Выводите статистику всех активных **TCP/IP**-подключений и **PID**-кодов процессов каждые 10 секунд.

11. Изучить статистику протокола и текущих соединений **TCP/IP** с использованием **NetBIOS over TCP/IP** на конкретных примерах:

1. Выведите таблицу имен **NetBIOS** физической рабочей станции;

2. Отобразите содержимое кэша имен **NetBIOS** собственного узла пользователя.

3. Выводите статистику сеанса **NetBIOS** по **IP**-адресу удаленного узла сети (192.168.220.13) через каждые 15 секунд.

3.3. Содержание отчета по лабораторной работе

Отчет по лабораторной работе оформляется в соответствии с требованиями государственного стандарта и должен содержать:

1) титульный лист;

2) описание и цель работы;

3) краткое описание служебных команд и утилит командной оболочки, предназначенных для построения и организации пакетных файлов и сценариев в среде ОС Windows XP;

4) результаты исследований работы служебных команд и утилит в соответствии с учебными заданиями лабораторной работы;

5) заполненные таблицы учебных заданий лабораторной работы (таблица 3.4).

Таблица 3.4. Результаты выполнения команды **Set**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		

6) алгоритмы, блок-схемы и тексты пакетных файлов;

7) выводы о проделанной работе.

4. Работа с ОС UNIX

Цель работы: изучить простейшие утилиты оболочки UNIX и ознакомление с языком базовых регулярных выражений и командой ***grep***.

Часть 1. Простейшие утилиты оболочки UNIX

Цель работы: изучить простейшие утилиты оболочки UNIX.

4.1. Краткие теоретические сведения

Основной функцией утилиты является перенос информации в пределах ОС. При рассмотрении каждой конкретной утилиты пользователя системы интересуют функции этой утилиты, а также ее имя, используемое для передачи в систему через пользовательский интерфейс в качестве команды для ОС. При работе с системой UNIX общий формат такой пользовательской команды следующий:

имя [флаги] [файлы],

где:

1) квадратные скобки заключают необязательную часть команды;

2) *имя* – пользовательское имя исполняемого файла, содержащего загрузочный модуль (машинный код) утилиты;

3) *файлы* – имена файлов, над которыми утилита выполняет свои действия. Различают *входные файлы*, информация из которых (или информация о которых) используется утилитой в качестве ее исходных данных, а также *выходные файлы,* в которые утилита помещает результаты своей работы. По умолчанию большинство системных утилит использует в качестве входного файла клавиатуру, а в качестве выходного файла – экран. Эти устройства (и соответствующие им файлы) часто называют соответственно *стандартным вводом* и *стандартным выводом;*

4) *флаги* – двоичные параметры команды, уточняющие действие, которое должна выполнить запускаемая утилита. Флаг задается своим именем из одной буквы, которой предшествует символ «-». Некоторые флаги уточняются своими параметрами, которые отделяются от имени флага пробелами.

Ниже приводится краткое описание утилит, используемых пользователями операционной системы UNIX для работы с файлами. После имени каждой утилиты в скобках приводится название аналогичной или близкой команды в MS-DOS. Рассматриваемые утилиты можно разбить на группы:

1) работа с файловой структурой системы;

2) создание каталогов и анализ их содержимого;

3) копирование, переименование и перенос файлов;

4) уничтожение файлов и каталогов;

5) работа с текстовой информацией;

6) поиск информации;

7) выдача справочной информации;

8) упрощение пользовательского интерфейса.

4.1.1. Утилиты для работы с файловой структурой системы

Первая важная задача, которую необходимо решить при изучении любого языка управления ОС, – освоение команд, предназначенных для работы с файловой структурой системы.

1. Вывод абсолютного имени текущего каталога (в MS-DOS отсутствует, так как это имя является частью приглашения к вводу команды):

pwd

Это наиболее простая команда UNIX, которая не имеет ни одного параметра. Как и другие команды, она вводится пользователем в ответ на приглашение UNIX (а точнее *shell)* в виде символов «$» или "#".

Пример

 # pwd <Enter>

/home/user #

где *<Enter>* — клавиша, нажатие которой завершает ввод текущей строки символов. Далее мы будем опускать запись этой клавиши, предполагая, что всякая командная строка, которой предшествует приглашение «#», должна завершаться нажатием этой клавиши.

2. Замена текущего каталога (в MS-DOS – *cd):*

cd *[каталог]*

Если каталог опущен, то текущим каталогом станет корневой каталог поддерева каталогов данного пользователя.

Имя каталога может быть как абсолютным, так и относительным. Если в начале относительного имени каталога записать символы «~/», то смещение нового текущего каталога вычисляется относительно корневого каталога данного пользователя. Если в качестве имени каталога задать символы «..», то новым текущим каталогом станет «родитель» действующего текущего каталога.

Данная утилита не имеет флагов. К этому добавим, что *cd*, вообще-то говоря, не является утилитой в полном смысле этого слова, так как она существует не в виде отдельного исполняемого файла, а в виде подпрограммы ОС (точнее ее интерпретатора команд). Подобное свойство обусловлено небольшими размерами данной подпрограммы и для пользователя не заметно.

Примеры:

а) *# pwd*
/home/user/11
 # cd
 # pwd
/home/user
#

б) *# pwd*
/home/user
 # cd 11
 # pwd
/home/user/11
#

в) *# pwd*
/home/user/11
 # cd ..
 # pwd
/home/user
#

г) *# pwd*
/home/user/11
cd ../..

 # pwd

/home

#

В примере (а) параметр у команды *cd* опущен. В этом случае текущим каталогом становится корневой каталог поддерева каталогов данного пользователя (в примере это каталог *user*). В примере (б) происходит «спуск» в подкаталог текущего каталога. В примере (в) происходит «подъем» на один уровень файловой структуры, а в примере (г) – сразу на два уровня.

Добавим, что задание абсолютного имени позволяет сделать текущим любой каталог. Это же можно сделать, используя в относительном имени символы «..». Относительное имя без этих символов позволяет задать только каталог, являющийся прямым потомком действующего текущего каталога.

3. Вывод содержимого каталога на экран (в MS-DOS – *dir*):

ls *[каталог или файлы]*

Если параметр опущен, то на экран выводится содержимое текущего каталога в алфавитном порядке, иначе – содержимое заданного каталога. Если заданы имена файлов, то на экран выводятся сведения об этих файлах, если их имена присутствуют в текущем каталоге.

Данная утилита имеет 23 флага. Приведем только некоторые из них:

1) *-R* – рекурсивный вывод подкаталогов заданного каталога;

2) *-F* – пометить исполняемые файлы символом «*», каталоги – символом «/», а символические связи – «@»;

3) *-l* – вывод наиболее подробной информации о файлах;

4) *-a* – вывод списка всех файлов и подкаталогов заданного каталога (по умолчанию имена, начинающиеся с символа «.», не выводятся).

В простейшем случае команда не содержит флагов и позволяет вывести лишь имена файлов (в том числе и подкаталогов).

Примеры:

а) *# ls*

a.txt b1 file prog

#

б) *# ls file*

file

#

Если в примере (б) текущий каталог не содержал бы запрашиваемый файл, то на экран было бы выведено сообщение об отсутствии требуемого файла.

4. Создание нового каталога (каталогов) (в MS-DOS – *mkdir*):

mkdir *каталоги*

Имена создаваемых каталогов могут быть заданы в любом виде: простые, относительные, абсолютные. Один из флагов данной утилиты:

-m – создать каталог с заданным режимом доступа.

Примеры:

а) *# ls*

1 22 prog

 # mkdir 333

 # ls

1 22 333 prog

#

б) *# pwd*

/home/user/111

 # ls ..

111 a fl

 # mkdir ../222 ../333

 # ls ..

111 222 333 a f1

#

5. Удаление каталогов:

1) удаление пустых каталогов (в MS-DOS – *rmdir):*

rmdir *каталоги*

Применение этой команды аналогично команде *mkdir*, хотя ее действие противоположно. Данная команда может уничтожить каталог только в том случае, если он не содержит файлов и подкаталогов;

2) удаление любых каталогов (в MS-DOS - *deltree):*

rm -r *каталоги*

Данная команда выполнит удаление заданного каталога и всех содержащихся в нем файлов и подкаталогов. Другие флаги этой команды:

-f – удаление файлов (подкаталогов) без запроса подтверждения;

-i – обязательный запрос подтверждения при удалении каждого файла (подкаталога).

Примеры:

а)
```
# ls
1 22 prog
# rm –r 1 prog
# ls
22
#
```

б)
```
# pwd
/home/user/111
# ls ..
111 a f1
# rm –r ../f1../a
# ls ..
111
#
```

Обратим внимание, что нельзя уничтожить каталог, если он является в данный момент текущим каталогом. Например, в примере (б) нельзя задать имя уничтожаемого каталога как «../111».

6. Копирование содержимого одного каталога в качестве содержимого другого каталога (в MS-DOS – *xcopy):*

cp -r *каталог_1 каталог 2*

В качестве первого параметра команды записывается имя каталога-источника, а в качестве второго параметра – имя каталога-приемника.

Копирование производится рекурсивно. То есть если каталог-источник имеет подкаталоги, то их содержимое также будет скопировано. При этом копирование каждого файла (подкаталога) означает создание на диске новой физической копии файла (подкаталога), имеющей такое же простое имя, что и копируемый файл (подкаталог). В том случае, если в каталоге-приемнике уже существует файл (подкаталог) с таким же простым именем, что и копируемый, то прежний файл (подкаталог) будет уничтожен.

4.1.2. Утилиты для работы с текстовой информацией

1. Вывод текстового файла на экран (в MS-DOS – *type*), создание файла с клавиатуры (в MS-DOS – *copy con*):

cat *[файлы]* – вывод текстового файла на экран
cat > *имя_файла* – создание файлы с клавиатуры

Данная утилита выводит на экран содержимое всех текстовых файлов, заданных в качестве ее параметров. При этом содержимое выводимых файлов на экране никак не разделяется. Если ни один из файлов не задан, то на экран выводится последовательность символов, введенная с клавиатуры (напомним, что клавиатура – тоже файл). Ввод с клавиатуры будет выполняться также в том случае, если вместо любого имени файла записан символ «–». Для завершения ввода символов с клавиатуры следует одновременно нажать две клавиши: <Ctrl>&<D> («конец файла»).

Примеры:

а) *# cat fa*
aaaaaaa
 # catfb
bbbbbbb
 # cat fc

б) *# cat fa - fb*
aaaaaaa
xxxxxxx
xxxxxxx
yyyyyyy

ccccccc
 # cat fa fb fc
aaaaaaa
bbbbbbb
ccccccc
#

yyyyyyy
<Ctrl>&<D>
bbbbbbb
#

В примере (а) сначала выводится содержимое файлов *fa*, *fb*, *fc* по отдельности, а затем вместе. В примере (б) на экран выводится то же содержимое файлов *fa* и *fb*, что и в примере (а). Между этими двумя выводами на экран выводится текст, набранный на клавиатуре. Данный текст состоит из двух строк: «*xxxxxxx*» и «*yyyyyyy*». После нажатия клавиши, соответствующей символу «*x*» или «*y*», мы тут же видим на экране его изображение – «эхо» символа. После того как набрана вся строка и нажата <*Enter*>, на экране опять появляется изображение этой строки. Это результат деятельности программы ***cat***, которая выполняет вывод строки, не дожидаясь завершения входного текста.

При создании файла, после того как набран весь текст, одновременно следует нажать две клавиши: *<Ctrl>&<D>*. Обратите внимание, что эти клавиши следует нажимать не в конце последней строки вводимого текста, а в начале следующей строки, то есть после нажатия <*Enter*>. Благодаря этому символ «конец строки» будет помещен в конец нашего файла (напомним, что клавиатура и экран – тоже файлы). Поэтому вывод на экран следующего файла (в примере это *fb)* начнется с новой строки.

2. Вывод строки символов на экран (в *MS-DOS – echo):*

echo *строка*

Как и команда ***cd***, данная команда выполняется не отдельной утилитой, а подпрограммой интерпретатора команд ОС.

Примеры:

а)
 # echo "Good morning!"
Good morning!
#

б)
 # echo Good morning
Good morning!
#

В примере (а) выводимая строка символов заключена в двойные кавычки, наличие которых требует воспринимать строку символов в качестве единого параметра команды *echo.* В примере (б) таких кавычек нет, и поэтому строка представляет собой два параметра. На результат выполнения команды *echo* подобное различие не влияет.

3. Вывод текста, вводимого с клавиатуры, на экран и одновременное копирование этого текста в заданный файл (файлы):

tee *файлы*

Один из флагов этой команды:

-a – запись текста не в начало файла (при этом файл создается заново), а его добавление в конец существующего файла (файлов).

Примеры:

а)
```
# cat f1
aaaaaaa
# cat f2
bbbbbbb
# tee f1 f2
ccccccc
ccccccc
<Ctri>&<D>
# cat f1 f2
ccccccc
ccccccc
#
```

б)
```
# cat fa
xxxxxx
# tee -a fa
yyyyyyy
yyyyyyy
<Ctr!>&<D>
# cat fa
xxxxxx
yyyyyyy
#
```

В примере (а) введенная с клавиатуры строка текста «*ccccccc*» записана в качестве нового содержимого файлов *fl* и *f2.* Наличие на экране двух таких строк обусловлено тем, что одна из них представляет собой «эхо» введенных символов, а вторая является одним из результатов выполнения команды *tee.*

В примере (б) задание в команде *tee* флага *-a* привело к тому, что введенная с клавиатуры строка была добавлена к прежнему содержимому файла *fa.*

4. Создание новых текстовых файлов и корректировка существующих. Данную функцию выполняют утилиты, называемые ***текстовыми редакторами.*** Примеры текстовых редакторов: *ed, ee, sed, vi* (текстовый редактор в *MS-DOS – edit).* В качестве примера приведем вызов редактора *sed:*

sed *[файлы]*

Данный редактор редактирует заданные в команде файлы построчно, от меньших номеров строк к большим, без возврата к ранее пройденным строкам. Редактирование строк производится согласно командам редактирования, заданным одним из двух способов:

1) в качестве параметров флага - ***e;***
2) команды редактирования содержатся в файле, имя которого задано в качестве параметра флага

Если ни одно имя файла в команде не задано, то по умолчанию входным файлом считается клавиатура. Набираемые на ней строки и будут подвергаться редактированию. В этом случае произойдет создание нового текстового файла, который с помощью интерпретатора команд ОС может быть записан на диск.

Более подробно редактор *sed* рассматривается в подразделе 5.2.

5. Сортировка и слияние файлов (в MS-DOS – *sort):*

sort *файлы*

Если флагов нет, то данная команда выполняет слияние перечисленных файлов в единый файл. Причем строки этого файла сортируются в лексографическом порядке. По умолчанию результат выводится на экран. Два флага этой команды:

-u – при наличии нескольких одинаковых строк результат содержит только одну строку;

-о файл – вывод результата делается не на экран, а в заданный файл.

6. Поиск файлов (в MS-DOS – *find*):

find *каталог [флаги]*

Данная утилита осуществляет поиск файлов в поддереве файловой структуры, корнем которого является заданный каталог. Условия поиска задаются с помощью флагов. В отличие от ранее перечисленных утилит флаги задаются в конце команды. Из всех многочисленных флагов обратим внимание на два:

1) -***type*** *mun* – поиск файлов указанного типа. Аргумент *mun* может принимать следующие значения: ***b*** (файл – блочное устройство), ***c*** (файл – символьное устройство), ***d*** (файл – каталог), ***f*** (обычный файл), ***l*** (файл – символическая связь), ***p*** (файл – именованный канал);

2) -***name*** *имя* – поиск файлов с указанным именем.

В одной команде *find* можно задать несколько условий поиска, соединив их при помощи следующих логических операторов:

-a – логическое И; ***-o*** – логическое ИЛИ; \! – логическое НЕ.

Примеры:

а)
```
# find ./ -name fa
./2/fa
./fa
# find ./ -name f9
#
```

б)
```
# find / -name find
/usr/share/irc/heip/note/find
#
```

В примере (а) осуществляется поиск файлов сначала с простым именем *fa*, а затем с именем *f9.* При этом найдено два файла с именем *fa* и ни одного с именем *f9.* В примере (б) задан поиск утилиты *find*, выполняющей рассматриваемую команду. В результате поиска на экран выведено абсолютное имя соответствующего файла.

В отличие от ранее рассмотренных команд, *find* имеет собственные метасимволы. ***Метасимвол*** – символ, имеющий для рассматриваемой команды специальное значение:

* – любая последовательность символов, в том числе пустая, за исключением символов «пробел» и «/»;

? – любой одиночный символ, за исключением «пробел»и «/»;

[...] — соответствует любому одиночному символу из тех, что перечислены через пробел в квадратных скобках. Пара символов, разделенных символом «–», соответствует одиночному символу, код которого попадает в диапазон между кодами указанных символов, включая их самих.

Примеры:

в) *# find / -name "sed*"*

/etc/setup/sed.ist.gz

/bin/sed.exe

/usr/info/sed.info

/usr/man/man1/sed.1

#

г) *# find /home -name 'f[1-3]'*

/home/111/1/2/f1

/home/111/1/3/f1

/home/111/1/12

/home/111/1/f3

#

В примере (в) осуществлен поиск по всей файловой структуре файлов, «родственных» текстовому редактору *sed*. В примере (г) осуществлен поиск в поддереве с корнем *home* файлов, простое имя каждого из которых состоит из двух символов: "*f*" и цифры от *1* до *3*.

Обратите внимание, что при использовании метасимволов имя файла обязательно заключено в кавычки (одиночные или двойные). Это объясняется тем, что *shell* имеет свои метасимволы, одноименные только что рассмотренным метасимволам *find.* Кавычки играют роль «экранирующих» символов, сообщая *shell* о том, что все символы, заключенные в кавычки, являются для *shell* обычными символами, которые должны быть переданы без изменений в запускаемую программу (в данном случае – в программу *find).*

Примеры:

д) *# find ./ -type d*

./

./ak3

./ak3/d6

./k4

е) *# find ./ -type d -a -name 'k*'*

./k4

./k4/k77

./k8

#

./k4/k77

./k8

./ab

#

В примере (д) выполняется поиск всех каталогов в поддереве, корнем которого является текущий каталог. Обратим внимание, что, в отличие от команды *ls,* вывод найденных каталогов производится не в алфавитном порядке. В примере (е) выводятся только те каталоги, простое имя которых начинается на букву *k*.

7. Поиск строк в текстовых файлах (в MS-DOS – *findstr*):

fgrep *подстрока [файлы]*

Данная утилита осуществляет поиск в перечисленных файлах строк, имеющих в своем составе шаблон – заданную подстроку. Найденные строки выводятся на экран. Если имена файлов опущены, то поиск осуществляется в тексте, вводимом с клавиатуры. При вводе с клавиатуры каждая строка, содержащая требуемую подстроку, повторяется дважды: первый раз она содержит «эхо» вводимых с клавиатуры символов, а второй раз выводится командой *fgrep.*

Некоторые флаги этой команды:

-x – выводятся только строки, полностью совпадающие с шаблоном;

-c – выводится только количество строк, содержащих шаблон;

-i – при поиске не различаются строчные и прописные буквы;

-l – выводятся только имена файлов, содержащих требуемые подстроки;

-n – перед каждой выводимой строкой записывается ее относительный номер в файле.

Если задан поиск в нескольких файлах, то перед выводом каждой строки выводится имя соответствующего файла. Заметим, что команда *fgrep* имеет нулевой код завершения в том случае, если найдена хотя бы одна строка, включающая заданную подстроку.

Примеры:

а) *# cat f3*
ODD
ddd

б) *# fgrep -i d f314*
f3: DDD
f3: ddd

HHH D
 # cat f4
tttt dd
DDD
 # fgrep d f3 f4
f3: ddd
f4: tttt dd
#

в) *# fgrep -/' h d' f3 f4*
f3: HHH D
#
#

f3: HHH D
f4: tttt dd
f4: DDD
#

г) *# fgrep -ic d f3 f4*
f3: 3
f4:2

Во всех четырех примерах поиск требуемых строк выполняется в файлах *f3* и *f4*, содержимое которых показано в примере (а). В этом примере выполнен поиск строк, содержащих символ «*d*». В примере (б) на экран выводятся строки, содержащие как символ «*d*», так и «*D*». В примере (в) производится поиск строк, содержащих подстроку «*h d*». Так как эта подстрока содержит пробел, то она обязательно должна быть заключена в кавычки (одиночные или двойные). Наличие кавычек сообщает *shell* о том, что набор символов между ними должен быть передан в программу *fgrep* как единый параметр. В примере (г) одновременно заданы два флага: *-i* и *-c*.

8. Выдача статистики о текстовых файлах (в MS-DOS отсутствует):

***wc** [файлы]*

Данная утилита выдает статистику о своих входных файлах. Если эти файлы не заданы, выдается статистика о тексте, введенном с клавиатуры.

Флаги этой команды:

-l – вывод числа строк;

-w – вывод числа слов;

-c – вывод числа символов.

По умолчанию все три флага установлены (*-lwc).* Поэтому флаги записываются в этой команде только тогда, когда требуется ограничить выходную статистику.

Примеры:

а) *# cat f1*

xx xx

yy yy

 # cat f2

zzzzz

 # wc f1 f2

2 4 14 f1

1 1 7 f2

3 5 21 total

#

б) *# wc -c*

This is a test to see if / am

entering text in the file "letter"

Once I have completed it I shall

that I have created 4 new lines of

<Ctrl>&<D>

145

#

В примере (а) выдается статистика о файлах *fl* и *f2*. Отсутствие флагов означает, что должна быть выведена полная статистика: число строк (первый столбец на экране), число слов (второй столбец) и число символов (третий столбец). Полученное число символов обусловлено тем, что в результате нажатия клавиши *<Enter>* в текстовую строку записывается не один, а два символа: «перевод строки» и «возврат каретки. В примере (б) выполнен подсчет символов, введенных с клавиатуры. Такая команда может быть использована, например, для определения скорости набора текста.

4.1.3. Утилиты для работы с файлами произвольного типа

В отличие от утилит, рассмотренных в предыдущем подразделе, данные утилиты предназначены для работы не только с текстовыми, но и с любыми другими файлами.

1. Копирование файла (в *MS-DOS – copy):*

ср *исходный файл конечный файл (или каталог)*

Примеры:

а)
```
# cat fa
aaaaaaa
# cat fb
bbbbbbb
# cp fa fb
# cat fb
aaaaaaa
#
```

б)
```
# ls 2
3 fx fy
# cp fa 2
# ls 2
3 fa fx fy
#
```

Следует отметить, что при любом копировании создается не новая жесткая связь (связи), а создается новый файл (файлы). Так, при копировании из файла в каталог в последнем создается новая запись, состоящая из простого имени исходного файла и из системного номера нового файла. При ранее рассматривавшемся копировании из каталога в каталог копируются все файлы (в том числе и подкаталоги) из исходного каталога в конечный каталог. При этом для каждого копируемого файла создается новый файл с точно таким же содержимым, после чего новый файл регистрируется в конечном каталоге.

В примере (а) файл *fa* копируется в тот же каталог, но под другим именем. Файл, существовавший прежде под этим именем, уничтожается. В примере (б) файл *fa* копируется в подкаталог 2 под своим именем.

2. Переименование файлов и их перемещение (в MS-DOS – *rename, move):*

mv *исходныйфайл (или каталог) конечный файл (или каталог)*

Если исходный и конечный файлы находятся в одном и том же каталоге, то данная утилита заменяет имя исходного файла на имя конечного файла. Если же эти файлы находятся в разных каталогах, то производится «перемещение» файла по файловой структуре системы. При этом запись файла в исходном каталоге уничтожается, а точно такая же запись в конечном каталоге, наоборот, создается. Если в качестве первого операнда задан файл, а в качестве второго – каталог, то также производится перемещение файла в заданный каталог. Если в качестве обоих

операндов заданы имена каталогов, то производится переименование каталога, соответствующего первому операнду.

Примеры:

а)
```
# ls
fa fb
# mv fb fx
# ls
fa fx
#
```

б)
```
# ls
2 5 fa
# ls 2
3 fx fy
# mv fa 2
# ls 2
3 fa fx fy
#
```

В примере (а) файл *fb* переименован в *fx.* А в примере (б) файл *fa* перемещен в подкаталог *2* текущего каталога.

3. Удаление файлов (в *MS-DOS – del):*

rm *файлы*

Это уже знакомая нам команда ***rm***, но без флага ***-r***. Параметрами команды являются имена удаляемых файлов. Другие флаги:

-f – удаление файлов без запроса подтверждения;

-i – обязательный запрос подтверждения при удалении каждого файла.

Следует отметить, что эта утилита удаляет не сами файлы, а записи о них в родительских каталогах. Само удаление файла происходит только в том случае, если число жестких связей для этого файла станет равным 0.

Примеры:

а)
```
# ls
fa fb fx fy
# rm fb fx
# ls
fa fy
#
```

б)
```
# ls
fa fb fx fy
# rm –i fb fx
rm: remove 'fb'? y
rm: remove 'fx'? n
# ls
fa fx fy
#
```

В примере (а) произведено удаление файлов *fb* и *fx* без запроса подтверждения на удаление файла. В примере (б) такое удаление произведено с запросами. При этом файл *fb* был удален, а *fx* – нет.

4. Создание жестких и символических связей (в MS-DOS отсутствует):

ln *исходный_файл файл_ссылка (или каталог)*

Эта команда создает новую связь с исходным файлом. При отсутствии флага *-s* создается жесткая связь с этим файлом. В этом случае файл-ссылка представляет собой новое имя уже существующего файла. Если в качестве второго параметра команды задано не имя файла, а имя каталога, то в этом каталоге исходный файл будет зарегистрирован под своим простым прежним именем. При наличии флага *-s* создаваемый файл-ссылка представляет собой символическую связь с исходным файлом.

4.1.4. Текстовый редактор *sed*

Это строковый пакетный редактор. Термин «строковый» означает, что редактирование текста производится построчно. При этом, выполнив преобразование текущей строки, нельзя в этом же сеансе работы редактора вернуться к редактированию одной из ранее пройденных строк. Термин «пакетный» означает, что редактирование очередной строки текста осуществляется путем выполнения пакета (списка) команд редактора *sed,* заданных при его запуске.

Мы будем использовать далее следующий формат команды *UNIX,* выполняющей запуск *sed:*

 # sed файл 1 . . . файл n -e {"
> команда 1
. . . . . .
> команда m
> "}

Данная команда *sed* выполняет редактирование текстовых файлов *1...n.* При этом сначала последовательно редактируются строки первого файла, затем строки второго файла и так далее до файла *n.* Вместо имени файла можно записать символ

«–», который означает, что часть исходного текста вводится с клавиатуры (вспомним, что в *UNIX* клавиатура также является файлом). Если имена файлов отсутствуют вовсе, то исходный текст полностью вводится с клавиатуры.

Флаг *-e* используется для задания пакета команд, предназначенного для редактирования каждой текстовой строки. Этот пакет заключен между символами '{'и '}', а также между символами ' " ' и ' " '. Обратим внимание, что каждая команда редактора расположена на отдельной строке. На отдельной строке находятся также завершающие символы команды ' "} '.

По умолчанию вывод отредактированного текста производится на экран. С помощью команд редактора *sed*, а также операций перенаправления вывода, выполняемых интерпретатором *shell*, можно обеспечить вывод результатов редактирования в требуемый файл на диске.

Как и в любой команде *shell,* занимающей более одной строки экрана, первая строка начинается с первичного приглашения *shell* в виде символа «*$*» или "*#*", а каждая последующая строка начинается с вторичного приглашения в виде символа «>».

Далее при рассмотрении команд редактора *sed* мы будем использовать два простых текстовых файла – *fl* и *f2:*

 # cat f1

1 22 333

4444 55555

 # cat f2

xxxxx

yyyyy

4.1.4.1. Команда вывода номера текущей строки: «=».

Если перед этой командой записать число, то будет пронумерована только строка, номер которой совпадает с этим числом. Если число отсутствует, то будут выведены номера всех строк.

Пример

 # sed f1 f2 -e {"

> =

> 2 =

>"}

1

1 22 333

2

2

4444 55555

3

xxxxx

4

yyyyy

#

В этом примере все строки обрабатываются первой командой «=», а строка *2* обрабатывается не только первой, но и второй командой. Поэтому она пронумерована дважды.

4.1.4.2. Команда добавления текста, вводимого с клавиатуры, после заданной строки:

a

текст

Как и для предыдущей команды, номер требуемой строки предшествует команде. Если он опущен, то вставка выполняется после каждой из строк.

Примеры:

а) *# sed f1 -e {"*

> a

> 88888

> 99999

> 1 a

> "}

1 22 333

8888899999

sssss

4444 55555

8888899999

#

б) *# sed f1 -e {"*

> a

> 88888

> 99999

> 1 a

> sssss

> "}

1 22 333

88888

99999

sssss

4444 55555

88888

99999

#

Оба примера выполняют добавление первого текста после каждой из введенных строк, а второго текста – только после строки *1*. Результат добавления первого из этих текстов в обоих примерах получился разный: в (а) это одна строка, а в (б) – две строки. Причиной такого различия является третья командная строка. В (а) она заканчивается одним символом «\», а в (б) – двумя. Кроме того, обратим внимание, что в обоих примерах команда *sed -a* заканчивается не одним, а двумя символами «\». Причиной обеих этих особенностей является та роль, которую играет символ «\» для *shell* и для *sed.*

Напомним, что все, что набрано после символа «*$*» или "#", представляет собой команду *shell* и поэтому подвергается его обработке. Эта обработка, в частности, заключается в том, что *shell* обрабатывает свои специальные символы (метасимволы), одним из которых является символ «\». Причем этот символ имеет для *shell* двойной смысл.

Во-первых, если «\» стоит в конце командной строки, то это означает, что на следующей строке пользователь набрал продолжение текущей *shell*-команды. Поэтому *shell* убирает из командной строки и символ «\» (он сделал свое дело), и символ «конец строки», который является результатом нажатия пользователем клавиши *<Enter>.*

Во-вторых, символ «\» рассматривается *shell* как «экранирующий» символ. Это означает, что если «\» записан в командной строке непосредственно перед метасимволом *shell*, например перед символом «*», то он утрачивает свое специальное значение (для «*» это подстановка любой символьной строки) и оставляется *shell* в командной строке как обычный алфавитно-цифровой символ. Так как «\» является метасимволом, то запись «\\» означает, что *shell* должен оставить «\» в командной строке как обычный символ.

В рассматриваемом примере используется еще одна пара «экранирующих» символов: «"» и «"». «Экранирующее» действие этих символов более слабое по сравнению с «\»: лишь некоторые метасимволы, заключенные между двойными кавычками, теряют свой специальный смысл. Например, символ «>» перестает быть метасимволом, а «\» им остается.

После того как *shell* сделает требуемые подстановки, он выполнит запуск исполняемого файла (у нас это *sed),* передав на вход запускаемой программы параметры командной строки, полученные *shell* в результате преобразования метасимволов. В рассматриваемом примере *shell* передаст на вход *sed* следующее:

а)
```
f1 -e {
a\
8888899999
1 a\
sssss
}
```

б)
```
f1 -e {
a\
88888\
99999
1 a\
sssss
}
```

Для *sed* символ «\» также является метасимволом, имеющим единственное значение: его запись в конце строки означает, что команда *sed* будет продолжена на следующей строке. В варианте (а) команда добавления *a* занимает две, а в варианте (б) – три строки. Заметим, что символы «{» и «}» также являются для *sed* специальными, обозначая начало и конец пакета команд.

4.1.4.3. Команда добавления текста, вводимого с клавиатуры, перед заданной строкой:

i
текст

Эта команда очень похожа на рассмотренную выше команду *a*. Единственное отличие: текст добавляется не после, а перед заданной строкой.

4.1.4.4. Замена строки или группы строк заданным текстом

c
текст

Перед командой *c* можно задать одно число или два числа, разделенных запятой. Если указано одно число, то будет заменена только одна строка с этим номером. Если указаны два числа, то будут заменены все строки в диапазоне номеров между этими числами включительно. Если числа перед командой вообще не заданы, то все исходные строки будут заменены заданным текстом. Остальные детали применения этой команды аналогичны командам *a* и *i*.

Примеры:

а)
```
# sed f1 f2 –e {"
> 2 c\\
> ooooooo\\
> uuuuuuu
"}
1 22 333
ooooooo
uuuuuuu
xxxxx
```

yyyyy

#

б) *# sed f1 f2 –e {"*

> 1,3 c

> ooooooo

> uuuuuuu

"}

ooooooo

uuuuuuu

yyyyy

#

4.1.4.5. Удаление заданных строк

Номера удаляемых строк задаются одним или двумя числами аналогично команде *c*. Если номера не заданы, то команда *d* удаляет все редактируемые строки.

Примеры:

а) *# sed f1 f2 –e {"*

> 1 d

> "}

4444 55555

xxxxx

yyyyy

#

б) *# sed f1 f2 –e {"*

> 1,3 d

>"}

yyyyy

#

4.1.4.6. Замена некоторой последовательности символов 1 на требуемую последовательность 2

s\последовательность 1\последовательность 2**pgw**

Модификаторы команды *p, g, w* необязательны и используются в следующих случаях:

p – редактируемая строка выводится (на экран или в файл) повторно в том случае, если требуемая замена успешно выполнена. Если команда вызова *sed* имеет дополнительный флаг -n, то выводятся только те строки, в которых произошла замена;

g – требуется выполнить замену столько раз, сколько заменяемая последовательность символов встретится в редактируемой строке. При отсутствии этого модификатора замена производится только один раз;

w – в случае успешной замены редактируемая строка записывается в файл, имя которого задается через пробел после этого модификатора.

Подобно командам *c* и *d*, команде *s* могут предшествовать одно или два числа, задающие строки, в которых выполняется замена. Если ни одно число не записано, то замена производится во всех строках.

Примеры:

а)
```
# sed f1 -e {"
> s\55\n\\
> "}
1 22 333
4444 n555
#
```

б)
```
# sed f1 -e {"
> s\55\n\g
> "}
1 22 333
4444 nn5
#
```

В примере (а) выполнена замена одной пары символов «55», а в примере (б) — замена двух пар. Обратим внимание, что в (а) вторая командная строка завершается двумя символами "\", потому что один из них используется для экранирования другого (см. команду *a*);

в)
```
# sed f1 -e {"
> s\4 5\a b c\p
> "}
1 22 333
444a b c5555
444a b c5555
#
```

г)
```
# sed f1 -n -e {"
> s\4 5\a b c\p
> "}
444a b c5555
#
```

В примере (в) выполняется добавочный вывод строки, в которой произошла замена. В примере (г) благодаря флагу *-n* и модификатору *p* выполняется вывод

только той строки, в которой произошла замена. Обратите внимание, что заменяемая и замещающая последовательности символов могут содержать пробелы;

д)
```
# sed f1 -e {"
> s\44\8\gw f3
> "}
1 22 333
88 55555
# cat f3
88 55555
#
```

е)
```
# sed f1 -n -e {"
> s\44\8\pgw f4
> "}
88 55555
# cat f4
88 55555
#
```

В примере (д) текст, выводимый на экран, отличается от текста, записываемого в файл. В примере (е) такого различия нет благодаря использованию модификатора *p* и флага *-n.* Обратите внимание, что в последнем примере используются все три модификатора команды *s.*

4.1.4.7. Запись редактируемых строк в файл

w файл

Эта команда (не путать с одноименным модификатором команды *s*) выполняет запись требуемых строк в заданный файл. Указание строк производится одним или двумя числами аналогично команде *c*. Если числа не заданы, в файл записываются все строки обрабатываемого текста.

Примеры:

а)
```
# sed f1 f2 -e {"
> 2,3 w f5
> s\4\A\g
> "}
1 22 333
AAAA 55555
xxxxx
yyyyy
```

#cat f5

4444 55555

xxxxx

#

б) *# sed f1 f2 -e {"*

> s\4\A\g

> 2,3 w f6

> "}

1 22 333

AAAA 55555

xxxxx

yyyyy

#cat f6

AAAA 55555

xxxxx

#

В примере (а) на экран выводятся все строки с выполненной заменой символа «*4*» на «*A*», но в файл *f5* записаны строки без подстановки. Это объясняется тем, что команда *w* предшествует команде *s*. Поменяв эти команды местами (пример (б)), получим файл *f6* с выполненными заменами. Это является иллюстрацией того факта, что каждая исходная строка обрабатывается командами *sed* в том порядке, в котором эти команды записаны.

Везде ранее мы обрабатывали исходный текст теми командами *sed*, которые тут же вводили с клавиатуры. То же самое можно сделать с помощью команд, заранее записанных в некоторый текстовый файл на диске. Подобный командный файл обрабатывается *sed* подобно тому, как скрипт обрабатывается *shell.* Для задания командного файла в команде вызова *sed* следует использовать флаг *-f*:

 # sed файл 1 ... файл n -f командный файл

#

При записи команд *sed* в командный файл следует учитывать, что они могут отличаться от соответствующих команд, вводимых с командной строки. Такое различие обусловлено тем, что командный файл для *sed* не обрабатывается *shell,* а

поступает непосредственно в *sed.* Поэтому символы «\», добавлявшиеся нами для экранирования от воздействия *shell,* теперь не нужны.

Примеры. Любой из рассмотренных ранее примеров можно переделать в пример по применению командных файлов. Сделаем это для: 1) пример (а) для пояснения команды ***a****;* 2) пример (б) для пояснения команды ***c***:

а)
```
# cat >s1
a\
88888\
99999
1 a\
sssss
<Ctrl>&<D>
# sed f1 –f s1
1 22 333
88888
99999
#
```

б)
```
# cat >s2
1,3 c\
ooooooo\
uuuuuuu
<Ctrl>&<D>
# sed f1 f2 –f s2
oooooooo
uuuuuuu
yyyyy
#
```

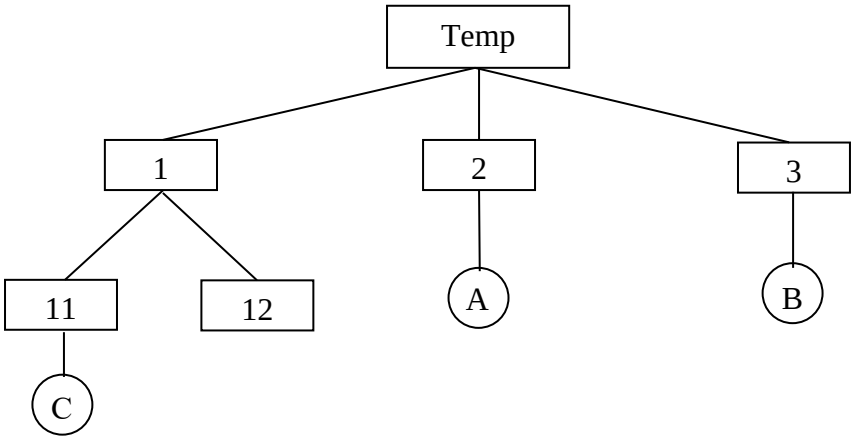
Допускается совместное применение флагов *-e* и *-f* в команде вызова *sed*. При этом сначала *sed* выполняет команды, введенные с клавиатуры, а затем команды из файла.

4.2. Подготовка к выполнению части 1 лабораторной работы

ВНИМАНИЕ!!! Если не выполнить написанное в этом абзаце, то невозможно будет выполнение всего данного раздела лабораторной работы!!! Запустите виртуальную машину с гостевой операционной системой SLAX Linux (выберите SLAX text mode). После загрузки гостевой операционной системы введите логин и пароль, указанные на экране.

4.3. Задание на часть 1 лабораторной работы

1. Создать дерево каталогов (имена каталогов и файлов должны быть такими, как указано; содержимое файлов может быть любым)



Примечание: В квадратиках – названия каталогов, в кружочках – файлов. Содержимое файлов – произвольное.

2. Создать в каталоге 2 файлы f1 и f2 со следующим содержимым

содержимое f1:

1 22 333

4444 55555

содержимое f2:

xxxxx

yyyyy

3. Вывести на экран файлы f1, f2

4. С использованием команды *sed* добавить (одновременно) текст в начало, в середину, в конец файла f1 (изменения будут только на экране, сохранение изменений в файле не требуется):

Перед первой строкой файла f1 вставить

666666

Между первой и второй строками:

7777777

После второй строки

88888888

5. Замена последовательности символов в файлах на другую (обратите внимание на разницу между параметром *w* и командой *w*):

5.1. В файле f1 заменить все встречающиеся символы 4 на символы A, результат (все измененные и неизмененные строки) сохранить в файле f3

5.2. В файле f2 заменить первый символ x на символ Z, результат (все измененные и неизмененные строки) сохранить в файле f4

5.3. В файле f2 заменить первый символ y на символ B, результат (все измененные и неизмененные строки) сохранить в файле f5

6. Вывод на экран файлов f3, f4, f5

7. Перемещение файлов, начинающихся с буквы f в каталог 12

8. Копировать в тот же каталог

Файл f1 – в f11

Файл f2 – в f12

Файл f3 – в f13

Файл f4 – в f24

Файл f5 – в f25

9. Переименовать файл f11 в файл a13

10. Копировать одной командой файлы в другой каталог – все файлы с именем, содержащим три символа, из которых первый – f, второй любой, а третий – 3,4 или 5 скопировать в каталог 11.

11. После проверки результатов выполнения предыдущих заданий удалить все созданные файлы и каталоги – удалить одной командой каталог Temp со всем содержимым.

Часть 2. Базовые регулярные выражения UNIX

Цель работы: ознакомление с языком базовых регулярных выражений и командой ***grep***.

4.4. Краткие теоретические сведения

Регулярные выражения представляют собой одно из наиболее интересных и полезных свойств операционной системы UNIX. Регулярные выражения являются языком описания текстовых шаблонов, который используется во многих системных утилитах для выполнения операций поиска и отбора при разнообразных обработках текстовых строк. Мы изучим регулярные выражения на примере применения их в утилите поиска *grep*.

Утилита поиска *grep*

Синтаксис

grep [опции] [шаблон] [файл...]

Описание

Команда *grep* выполняет поиск строк, соответствующих шаблону, заданному регулярным выражением, в файлах или во входном потоке. Если команда задана без опций, выводятся все найденные строки. Если имя файла не задано, команда выполняет поиск во входном потоке. Если задано несколько имен файлов или в составе имени файла использован символ '*', *grep* перед строкой выводит имя файла, которому эта строка принадлежит.

Опции

-c	выводится только число строк файла, соответствующих шаблону
-f *файл*	чтение шаблона из *файла*
-h	не выводятся имена файлов, в которых найдены строки, соответствующие шаблону
-i	игнорирование верхнего/нижнего регистров
-l	выводятся только имена файлов, в которых найдены строки, соответствующие шаблону

-n	перед каждой выводимой строкой выводится ее номер в файле
-v	ищутся строки, не соответствующие заданному шаблону
-w	ищутся слова, полностью соответствующие шаблону
-x	ищутся строки, полностью соответствующие шаблон

Регулярные выражения

Регулярные выражения представляют собой язык описания текстовых шаблонов. Регулярные выражения содержат образцы символов, входящих в искомое текстовое выражение, и конструкции, определяемые специальными символами (метасимволами).

Метасимволы, используемые в регулярных выражениях

^	Если необходимо выбрать первые символы строки (ставится перед выражением)
$	Если необходимо выбрать последние символы строки (ставится после выражения)
[]	любой символ, заключенный в квадратные скобки; чтобы задать диапазон символов, в квадратных скобках указываются через дефис первый и последний символы диапазона
[^]	любой символ, кроме символов, заданных в квадратных скобках
.	любой отдельный символ
\	отменяет специальное значение следующего за ним метасимвола
*	указывает, что предыдущий шаблон встречается 0 или более раз
\{n\}	указывает, что предыдущий шаблон встречается ровно n раз
\{n,\}	указывает, что предыдущий шаблон встречается не менее n раз
\{,n\}	указывает, что предыдущий шаблон встречается не более n раз
\{n,m\}	указывает, что предыдущий шаблон встречается не менее n и не более m раз

Примеры регулярных выражений

^the	ищутся строки, начинающиеся с буквосочетания "the"
be$	ищутся строки, заканчивающиеся буквосочетанием "be"
[Ss]igna[lL]	ищутся строки, содержащие буквосочетания: "signal", "Signal", "signaL" или "SignaL"
\.	ищутся строки, содержащие точку
^...th	ищутся строки, содержащие символы "th" в 4-й и 5-й позициях
^.\{53\}th	ищутся строки, содержащие символы "th" в 54-й и 55-й позициях
^.\{10,30\}th	ищутся строки, содержащие символы "th" в любых позициях между 10й и 30-й включительно
^.....$	ищутся строки, состоящие из 5 любых символов
^t.*e$	ищутся строки, начинающиеся с буквы "t" и заканчивающиеся буквой "e"
[0-9][a-z]	ищутся строки, содержащие комбинацию: цифра-прописная буква
[^123]	ищутся строки, не содержащие цифр "1" или "2" или "3"

Особые указания

Регулярное выражение берется в одинарные кавычки ('регулярное_выражение').

Перед знаками фигурных скобок { и } всегда ставится знак слэша "\", чтобы оболочка воспринимала их именно как служебный символ, внутри которого указывается какое-то условие.

Чтобы пропустить известное фиксированное количество любых символов с начала строки, выражение включает

^.{число_символов}

Чтобы пропустить известное меняющееся количество любых символов с начала строки, выражение включает

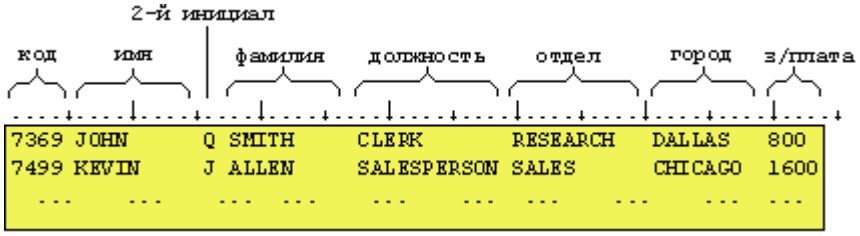
^.{минимальное_количество_символов,максимальное_количество_символов}

Чтобы выбрать любое число, выражение включает [0-9]

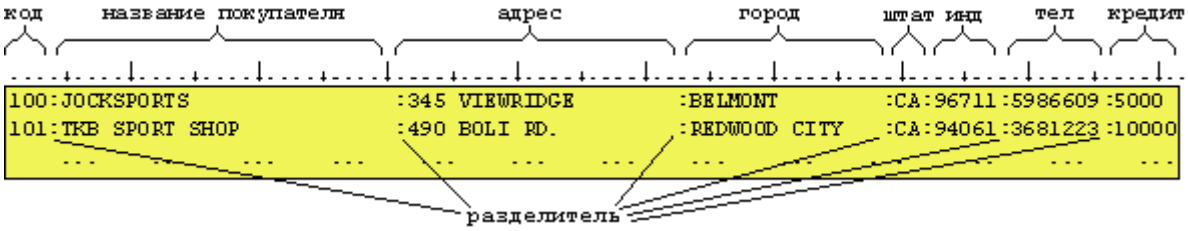
4.5. Структура файлов query 1 – query 5

Для выполнения лабораторной работы необходимо знать структуру файлов, из которых осуществляется выборка данных. Ниже представлена структура всех файлов, необходимых для данной практической работы.

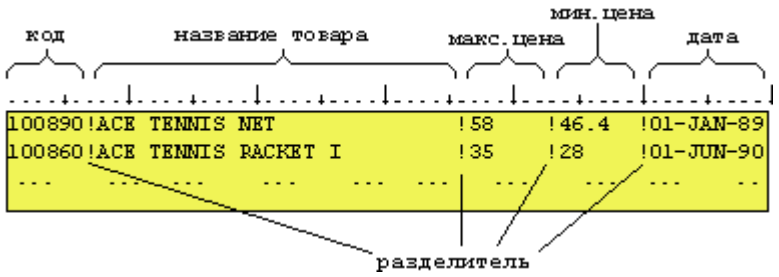
Структура файла query1



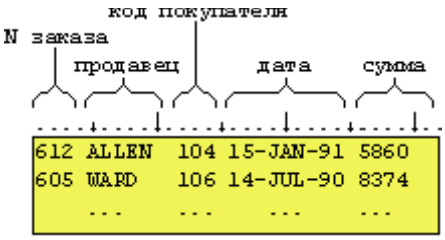
Структура файла query2



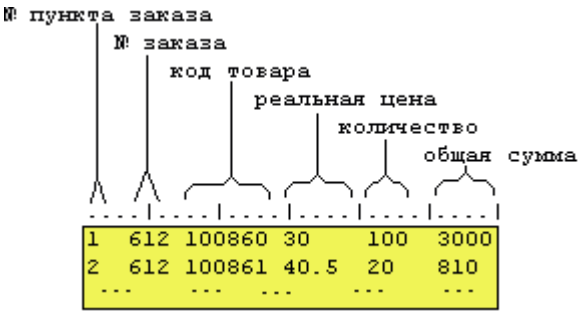
Структура файла query3



Структура файла query4



Структура файла query5



4.6. Подготовка к выполнению части 2 лабораторной работы

ВНИМАНИЕ!!! Если не выполнить написанное в этом абзаце, то невозможно будет выполнение всей практической работы!!! Запустите виртуальную машину с гостевой операционной системой SLAX Linux (выберите SLAX graphics VESA mode). После загрузки гостевой операционной системы, на рабочем столе дважды щелкните по ярлыку **System**. В открывшемся окне выберите **Storage Media**, затем – **Floppy Drive**. Когда увидите содержимое виртуальной дискеты, закройте это окно.

4.7. Задание на часть 2 лабораторной работы.

1. Откройте консоль (второй значок в виде черного прямоугольника с белой окантовкой слева внизу), перейдите в корневой каталог, откуда зайдите в подкаталог **disk** каталога **media**, где убедитесь в наличии в нем файлов с именами query 1 – query 5. Все задания необходимо выполнять, находясь в этом каталоге.

2. В файле query1 выбрать сотрудников, имеющих должность "менеджер" или "клерк" (по выбору).

3. В файле query1 выбрать все строки, в которых зарплата сотрудников равна круглому числу тысяч.

4. В файле query2 выбрать все строки, в которых в адресе имеется обозначение улицы (ST.)

5. В файле query3 выбрать все строки, в которых максимальная цена равна 20.

6. В файле query4 выбрать все строки, в которых дата продажи - весна 1990 г.

7. В файле query5 выбрать все строки, в которых указан товар, имеющий код 100860 и проданный по цене 35

4.8. Содержание отчета по лабораторной работе

Внимание!!! При выполнении каждого из заданий обеих частей используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 4.1.

Таблица 4.1. Результаты выполнения команд

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		
6.		

Отчет по лабораторной работе оформляется в соответствии с требованиями государственного стандарта и должен содержать:

1) титульный лист;

2) описание и цель работы;

3) краткое описание служебных команд и утилит, предназначенных для работы с файлами и дисками в среде командной оболочки;

4) результаты исследований работы служебных команд и утилит в соответствии с учебными заданиями лабораторной работы;

5) заполненные таблицы учебных заданий лабораторной работы;

6) выводы о проделанной работе.

ЗАКЛЮЧЕНИЕ

Методические указания для выполнения лабораторных работ по дисциплине "Операционные системы ЭВМ" позволяет изучить практические основы настройки и администрирования операционных систем Windows семейства NT и UNIX. После выполнения всех лабораторных и практических работ студенты получают достаточный уровень в области работы с операционными системами.

ЛИТЕРАТУРА

1. Танненбаум Э. Современные операционные системы. 3-е изд. – СПб.:Питер, 2010. – 1120 с.: ил. – (Серия "Классика Computer Science").

2. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. – СПб.: Питер, 2002. – 544 с.: ил.

3. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс. / Пер. с англ. – 4-е изд. – М.: Издательско-торговый дом "Русская Редакция"; СПб.: Питер, 2005. – 992 с.: ил.

4. Пахмурин Д.О. Операционные системы ЭВМ: Учебное пособие. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2013. – 254 с.: ил.