Приложение
к рабочей программе
учебной дисциплины

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для реализации программы дисциплины

ОП.01 Операционные системы и среды

для специальности
09.02.07 Информационные системы и программирование

Базовая подготовка

2023

СОДЕРЖАНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Дисциплина ЕН.02. Информатика по заочной форме обучения изучается на 1 курсе по специальности 09.02.07 Информационные системы и программирование.

Для закрепления теоретических знаний и отработки практических навыков программой дисциплины предусмотрены практические занятия. Практические занятия проводятся в период лабораторно-экзаменационных сессий под руководством преподавателя.

В конце методических рекомендаций помещены вопросы для самопроверки при подготовке к дифференцированному зачету и список рекомендуемой литературы.

Цель контрольной работы – закрепить знания, полученные обучающимся при самостоятельном изучении дисциплины по рекомендуемой литературе.

После проверки работы преподавателем, обучающийся должен выполнить работу над ошибками (если они имеются в работе). Работа над ошибками выполняется на том же электронном носителе и сдается на повторную проверку.

Комплект заданий для выполнения
практических работ

Практическая работа №1
Диспетчер задач Windows.
Цель работы: Практическое знакомство с управлением вводом/выводом в операционных системах Windows и кэширования операций ввода/вывода.

План проведения занятия:

1. Ознакомиться с краткими теоретическими сведениями.
2. Ознакомиться с назначением и основными функциями Диспетчера задач Windows.
3. Приобрести навыки применения командной строки Windows. Научиться запускать останавливать и проверять работу процессов.
4. Сделать выводы о взаимосвязи запушенных процессов и оперативной памятью компьютера.
5. Подготовить отчет о выполнении практической работы

Оборудование:
Аппаратная часть: персональный компьютер, сетевой или локальный принтер.
Программная часть: ОС Windows 7, текстовый процессор Microsoft Word.

Краткие теоретические сведения:
Необходимость обеспечить программам возможность осуществлять обмен данными с внешними устройствами и при этом не включать в каждую двоичную программу соответствующий двоичный код, осуществляющий собственно управление устройствами ввода/вывода, привела разработчиков к созданию системного программного обеспечения и, в частности, самих операционных систем.
Программирование задач управления вводом/выводом является наиболее сложным и трудоемким, требующим очень высокой квалификации. Поэтому код, позволяющий осуществлять операции ввода/вывода, стали оформлять в виде системных библиотечных процедур; потом его стали включать не в системы программирования, а в операционную систему с тем, чтобы в каждую отдельно взятую программу его не вставлять, а только позволить обращаться к такому коду. Системы программирования стали генерировать обращения к этому системному коду ввода/вывода и осуществлять только подготовку к собственно операциям ввода/вывода, то есть автоматизировать преобразование данных к соответствующему формату, понятному устройствам, избавляя прикладных программистов от этой сложной и трудоемкой работы. Другими словами, системы программирования вставляют в машинный код необходимые библиотечные подпрограммы ввода/вывода и обращения к тем

системным программным модулям, которые, собственно, и управляют операциями обмена между оперативной памятью и внешними устройствами.
Таким образом, управление вводом/выводом — это одна из основных функций любой ОС.
Одним из средств правления вводом/выводом, а также инструментом управления памятью является диспетчер задач Windows, он отображает приложения, процессы и службы, которые в текущий момент запущены на компьютере. С его помощью можно контролировать производительность компьютера или завершать работу приложений, которые не отвечают.
При наличии подключения к сети можно также просматривать состояние сети и параметры ее работы. Если к компьютеру подключились несколько пользователей, можно увидеть их имена, какие задачи они выполняют, а также отправить им сообщение.
Также управлять процессами можно и «вручную» при помощи командной строки.
Команды Windows для работы с процессами:

- at — запуск программ в заданное время
- Schtasks — настраивает выполнение команд по расписанию
- Start — запускает определенную программу или команду в отдельном окне.
- Taskkill — завершает процесс
- Tasklist — выводит информацию о работающих процессах

  Для получения более подробной информации, можно использовать центр справки и поддержки или команду help (например: help at)

- cmd.exe — запуск командной оболочки Windows

Ход работы:

Задание 1. Работа с Диспетчером задач Windows 7.

1. Запустите Windows 7
2. Запуск диспетчера задач можно осуществить двумя способами:

   1) Нажатием сочетания клавиш Ctrl+Alt+Del. При использовании данной команды не стоит пренебрегать последовательностью клавиш. Появится меню, в котором курсором следует выбрать пункт «Диспетчер задач».
   2) Переведите курсор на область с показаниями системной даты и времени и нажмите правый клик, будет выведено меню, в котором следует выбрать «Диспетчер задач».

3. Будет выведено окно как на рис. 1.

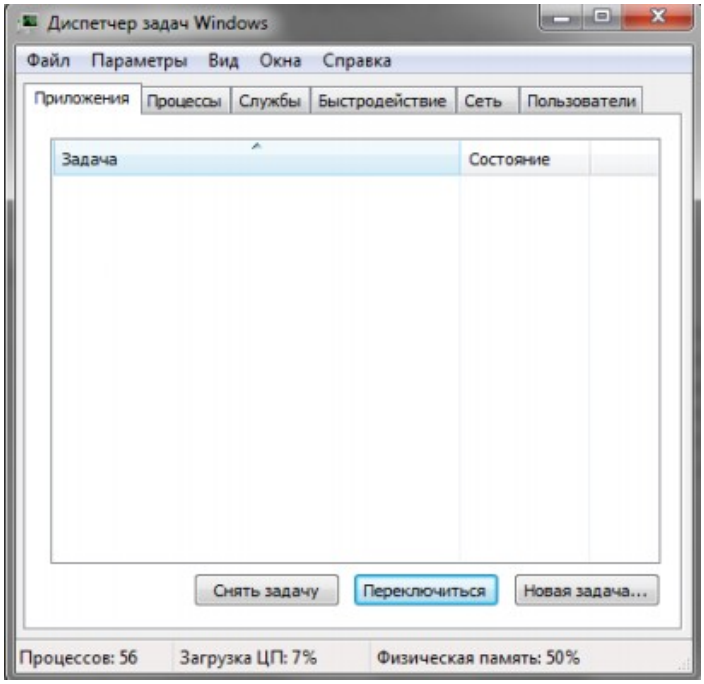


Рис. 1. Диспетчер задач Windows 7.

В диспетчере задач есть 6 вкладок:

1. Приложения
2. Процессы
3. Службы
4. Быстродействие
5. Сеть
6. Пользователи

- Вкладка «Приложения» отображает список запущенных задач (программ) выполняющиеся в настоящий момент не в фоновом режиме, а также отображает их состояние. Также в данном окне можно снять задачу переключиться между задачами и запустить новую задачу при помощи соответствующих кнопок.
- Вкладка «Процессы» отображает список запущенных процессов, имя пользователя запустившего процесс, загрузку центрального процессора в процентном соотношении, а также объем памяти используемого для выполнения процесса. Также присутствует возможность отображать процессы всех пользователей, либо принудительного завершения процесса. Процесс — выполнение пассивных инструкций компьютерной программы на процессоре ЭВМ.
- Вкладка «Службы» показывает, какие службы запущены на компьютере. Службы
- — приложения, автоматически запускаемые системой при запуске ОС Windows и выполняющиеся вне зависимости от статуса пользователя.
- Вкладка «Быстродействие» отображает в графическом режиме загрузку процессора, а также хронологию использования физической памяти компьютера. Очень эффективным

инструментом наблюдения является «Монитор ресурсов». С его помощью можно наглядно наблюдать за каждой из сторон «жизни» компьютера. Подробное изучение инструмента произвести самостоятельно, интуитивно.

- Вкладка «Сеть» отображает подключенные сетевые адаптеры, а также сетевую активность.
- Вкладка «Пользователи» отображает список подключенных пользователей.
- Потренируйтесь в завершении и повторном запуске процессов.
- Разберите мониторинг загрузки и использование памяти.
- Попытайтесь запустить новые процессы при помощи диспетчера, для этого можно использовать команды: cmd, msconfig.

5. После изучения диспетчера задач:

Задание 2. Командная строка Windows.

1. Для запуска командной строки в режиме Windows следует нажать:

(Пуск) > «Все программы» > «Стандартные» > «Командная строка»

2. Поработайте выполнением основных команд работы с процессами: запуская, отслеживая и завершая процессы.

Основные команды
Schtasks — выводит выполнение команд по расписанию
Start — запускает определенную программу или команду в отдельном окне. Taskkill — завершает процесс
Tasklist — выводит информацию о работающих процессах

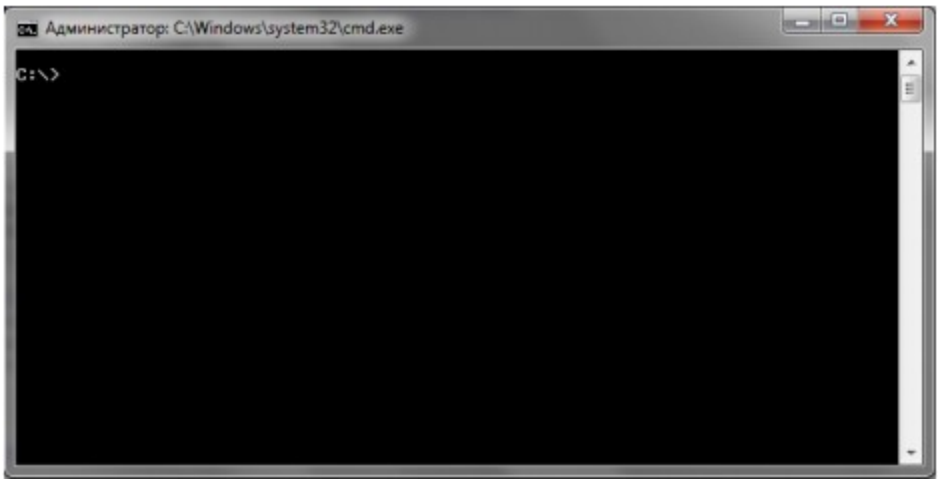


Рис. 2. Командная строка Windows 7.

3. В появившемся окне (рис. 2) наберите:

cd\ — переход в корневой каталог;
cd windows – переход в каталог Windows.
dir — просмотр содержимого каталога.
В данном каталоге мы можем работать с такими программами как «WordPad» и «Блокнот».

4. Запустим программу «Блокнот»:

   C:\Windows > start notepad.exe
   Отследим выполнение процесса: C:\Windows > tasklist

   Затем завершите выполнение процесса: C:\Windows > taskkill /IM notepad.exe

5. Самостоятельно, интуитивно, найдите команду запуска программы WordPad.

   Необходимый файл запуска найдите в папке Windows.

6. Выполнение задания включить в отчет по выполнению лабораторной работы.

Задание 3. Самостоятельное задание.

1. Отследите выполнение процесса explorer.exe при помощи диспетчера задач и командной строки.
2. Продемонстрируйте преподавателю завершение и повторный запуск процесса explorer.exe из:
   - Диспетчера задач;
   - Командной строки.
3. Выполнение задания включить в отчет по выполнению лабораторной работы.

Контрольные вопросы:

1. Дайте понятие процессу в операционной системе.
2. Дайте понятие службе в операционной системе.
3. Причислите основные команда работы с процессами при помощи командной строки.

Практическая работа № 2
Файловые системы ОС Windows

Цель работы: Целью работы является приобретение навыков анализа физической и логической структуры магнитных дисков и закрепление знаний по файловым системам *FAT* и *NTFS*.

Методические указания

На физическом уровне работа с устройством внешней памяти всегда проводится через соответствующий контроллер. Именно контроллер знает физические параметры устройства, поэтому физическая модель магнитных дисков не зависит от операционной системы (*Linux* или *Windows*).

Диск *Windows* и *Linux*, может быть разбит на несколько разделов, которые могут быть первичными, расширенными или логическими. Параметры разделов хранятся в таблице разделов диска (*partition table*). В отличие от *Linux*, разделы диска называются логическими дисками и обозначаются латинскими буквами, за которыми следует двоеточие – *A*:, *B*:, *C*:, *D*: и т.д. Файловые системы каждого раздела, в отличие от *Linux*, не связаны между собой и функционируют отдельно друг от друга.

Небольшая часть адресного пространства каждого раздела выделяется под системную область, основная часть – под область данных. Единицей дисковой памяти в области данных является кластер, размер которого кратен размеру сектора и может достигать 64 Кбайт. Все кластеры имеют сквозную нумерацию, причем первый допустимый номер кластера равен 2. Файлы на диске могут храниться в несмежных кластерах, т.е. в различных частях диска.

Файл *Windows* – это поименованная совокупность информации, хранящаяся на ВЗУ. В виде файлов на диске хранятся программы и данные. Каталог – файл специального формата, предназначенный для хранения метаданных о зарегистрированных в этом каталоге файлах и подкаталогах (имя, расширение, атрибуты, размер, дата и время создания или последнего изменения, адрес). Каталоги каждого логического диска организованы в единую древовидную структуру. Имена файлов и каталогов, в отличие от *Linux*, регистронезависимы.

Структура системной области диска зависит от типа файловой системы, которая определяет способы доступа к информации в области данных. Наиболее известными файловыми системами для *Windows* являются *FAT* и *NTFS*.

Файловая система FAT

На рис. 1 приведена логическая модель диска с файловой системой FAT. В системной области находятся загрузочная запись, таблица размещения файлов и корневой каталог. Загрузочная запись, иногда называемая начальным загрузчиком, имеет размер 512 байт, всегда хранится в нулевом секторе и используется в процессе загрузки операционной системы.

Рис. 1. Структура логического диска *FAT*

Таблица размещения файлов, в оригинальной литературе называемая *FAT* (*File Allocation Table*), содержит информацию о размещении файлов в области данных. Она всегда

занимает сектора, начиная с первого. На любом диске для обеспечения надежного доступа к данным всегда хранится две копии *FAT*, которые обновляются одновременно.

Таблица размещения файлов содержит информацию о номерах кластеров, выделенных для хранения каждого файла. Она представляет собой карту (образ) области данных, в которой описывается состояние каждого кластера диска. Размер таблицы зависит от объема диска. Номер начального кластера, выделенного файлу, записывается в элемент каталога этого файла.

Каждый элемент таблицы соответствует одному кластеру в области данных. Дефектные кластеры помечаются как "*bad*". Если кластер свободен, то соответствующий ему элемент *FAT* имеет значение "0". Если кластер выделен для какого-либо файла, то возможны два варианта:

- элемент содержит признак конца файла "*EOF*", если этот кластер является последним кластером, выделенным файлу;
- элемент содержит значение номера следующего кластера, выделенного файлу.

Таким образом элементы *FAT*, выделенные одному файлу, связываются в цепочки, позволяющие получить доступ к информации даже в том случае, если файл при записи разбивается на несколько фрагментов, хранящихся в несмежных кластерах диска. Элементы *FAT* могут быть 16- и 32-разрядными, в зависимости от этого файловые системы имеют названия *FAT*-16 и *FAT*-32 и работают с 16-разрядными и 32-разрядными дисковыми адресами соответственно.

Логическое разбиение области данных на кластеры, как совокупность секторов, взамен использования одиночных секторов имеет следующий смысл:

- уменьшается размер *FAT*;
- уменьшается возможная фрагментация файлов;
- ускоряется доступ к файлу, т.к. в несколько раз сокращается длина цепочек фрагментов дискового пространства, выделенных для него

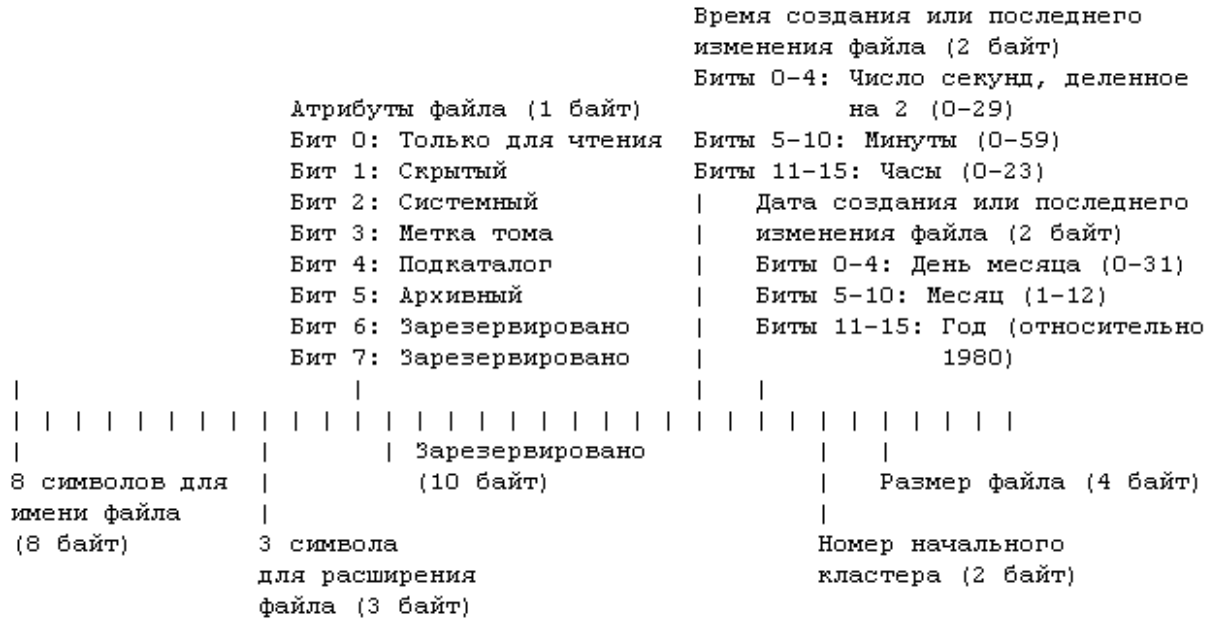
Следует иметь ввиду, что при увеличении размера кластера ухудшается коэффициент использования дисковой памяти за счет увеличения внутренней фрагментации. Минимальный размер кластера на диске с файловой системой FAT зависит от объема диска ($V_{диска}$) и разрядности элемента *FAT* (*r*): $V_{кл\_\min} = \frac{V_{диска}}{2^r}$

Корневой каталог – главный каталог диска, который занимает сектора, следующие за *FAT*. Фиксированное число элементов и размещение в системной области корневого каталога являются принципиальным отличием от прочих каталогов.

Записи корневого каталога имеют длину 32 байта, структура записей представлена в табл. 1. Если файл не имеет расширения, то в соответствующем поле хранятся пробелы. Дата и время используются в виде четырехбайтового значения в операциях сравнения. Номер начального кластера определяет точку входа в *FAT* для данного файла и одновременно дисковый адрес собственно файла.

- Размер элемента каталога – 32 байта

Содержание	Размер (байт)
Имя файла	8
Расширение	3
Байт атрибутов	1
Зарезервировано	10
Время	2
Дата	2
Номер начального кластера	2
Размер файла	4



На рис.2 представлена схема работы и организации *FAT*, а также фрагментация, когда части файла разбросаны по всему диску.

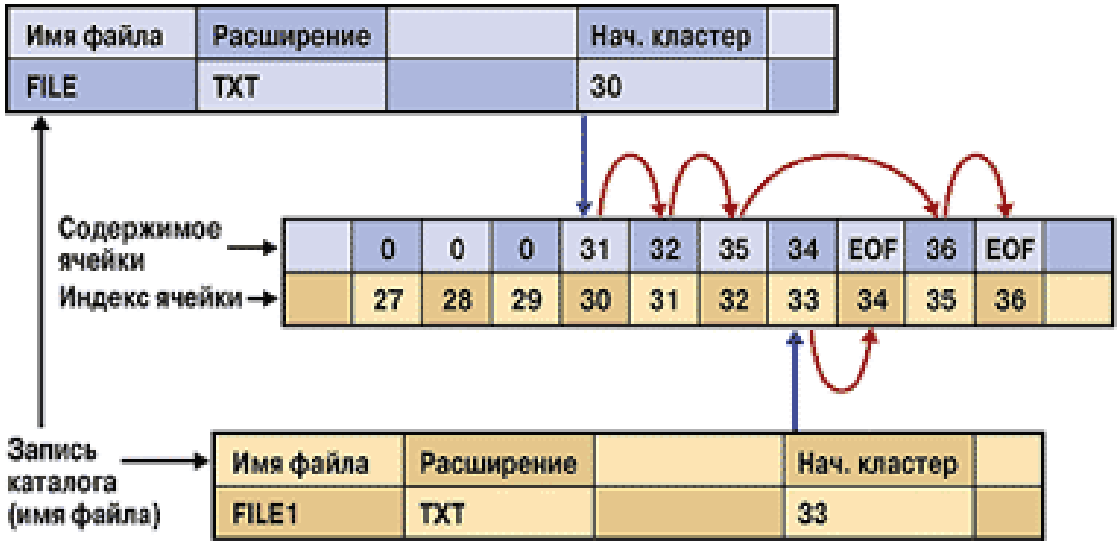


Рис.2. Схема работы и организации *FAT*

Цепочка кластеров для файла *FILE.TXT*: **30, 31, 32, 35, 36, *EOF***
Цепочка кластеров для файла *FILE*1.*TXT*: **33, 34, *EOF***

Файл, который занимает на диске более одного непрерывного участка, называется фрагментированным.

Фрагментация диска – это появление на диске множества свободных участков, разделенных занятыми участками.

Дефрагментация диска – это перемещение данных на разделе, после которого, кластеры содержащие части одного файла, размещаются последовательно.

Файловые системы *FAT* выделяют для записываемых на диск файлов некоторое количество кластеров, в зависимости от размера файла. В процессе работы с диском при записи и удалении файлов разного размера на диске появятся свободные и занятые области разной длины.

Такой метод хранения файлов позволяет использовать всё имеющееся на диске свободное место, т.к. если длина записываемого файла больше, чем размеры непрерывных свободных участков, то файл просто расположится в нескольких несмежных участках.

Реально время чтения сильно фрагментированного файла по сравнению с файлом, занимающим непрерывную область на диске, может отличаться в несколько раз! Внешне это выглядит так, как будто все программы стали работать в несколько раз медленнее, при этом наблюдается интенсивное перемещение головок диска от одного участка файла к другому.

Удаление файлов

При удалении файла обычно выполняются следующие действия:

- в таблице размещения файлов обнуляются все элементы, выделенные для описания этого файла;
- в соответствующем элементе каталога изменяется имя файла – вместо первого символа в поле имени записывается символ «*x*».

Остальные характеристики файла в элементе каталога, а также содержимое файла в кластерах диска, не изменяются, поэтому всегда есть возможность полностью или частично восстановить удаленный файл.

Полное восстановление возможно, если:

- не перезаписан соответствующий элемент каталога;
- имеется доступ к каталогу;
- кластеры, ранее занимаемые файлом, не выделены другим файлам или каталогам;
- удаленный файл был нефрагментированным.

При несоблюдении последнего условия полное восстановление не гарантируется, т.к. не всегда возможно извлечь данные о том, какие кластеры были выделены файлу.

Современные операционные системы обычно проводят удаление файлов в специальный скрытый каталог, который называется корзиной. Размер корзины может устанавливаться пользователем. Корзина обслуживается специальной программой, что делает восстановление ошибочно удаленных файлов удобным и быстрым.

Файловая система *NTFS*

Организация раздела NTFS

Как и любая другая система, *NTFS* делит все полезное место на кластеры – блоки данных, используемые единовременно. *NTFS* поддерживает почти любые размеры кластеров – от 512 байт до 64 Кбайт, неким стандартом же считается кластер размером 4 Кбайт.

Диск *NTFS* условно делится на две части. Первые 12% диска отводятся под так называемую *MFT* зону – пространство, в которое растет метафайл. Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой – это делается для того, чтобы самый главный, служебный файл (*MFT*) не фрагментировался при своем росте. Остальные 88% диска представляют собой обычное пространство для хранения файлов (рис.3).

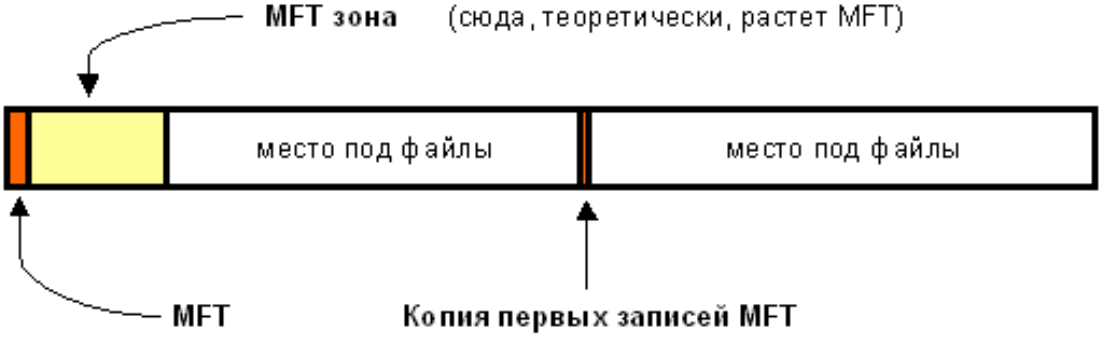


Рис. 3. Структура раздела *NTFS*

Основной структурой данных в *NTFS* является главная таблица файлов (*Master File Table, MFT*), которая хранится в системном файле *$MFT* и представляет собой главный каталог, в котором регистрируются все файлы раздела, включая системные файлы. Для *MFT* резервируется 12% от общего объема раздела в виде непрерывной последовательности блоков, которая называется *MFT*-зоной. Запись файлов и каталогов в эту зону не проводится, а ее адрес хранится в загрузочной записи.

MFT состоит из множества записей размером 1 Кбайт о файлах, расположенных на томе. В записи *MFT* хранится вся информация о файле (имя, дата и время создания, размер, положение на диске отдельных фрагментов, и т. д). Если не хватает одной записи *MFT*, то используются несколько, причем не обязательно подряд. При этом первая запись называется базовой. Каждая запись *MFT* имеет уникальный номер – индекс, общее количество записей – до 2^{48}.

Первые 16 записей файла *$MFT* выделены для хранения информации о системных файлах. Самая первая запись в *MFT* – это запись о самом файле *$MFT*. Во второй записи содержится информация о зеркальной копии *MFT* (файл *$MFTMirr*), в которой дублируются первые 4 записи таблицы *MFT*. В случае возникновения сбоя, если *MFT* окажется недоступным, информация о системных файлах будет считываться из файла *$MFTMirr*, адрес которого также имеется в загрузочной записи.

Ниже приведено назначение некоторых системных файлов *NTFS*:

- *$LogFile* – файл журнала, в котором записывается информация о всех операциях, изменяющих структуру раздела *NTFS*, например, создание файлов и каталогов. Файл журнала используется при восстановлении тома *NTFS* после сбоев;
- *$Volume* – файл информации о томе, в котором содержатся имя тома (Volume label), версия NTFS и набор флагов состояния тома, например, флаг, установка которого означает, что том был поврежден и требует восстановления при помощи системной утилиты Chkdsk;
- *$AttrDef* – таблица определения атрибутов, содержащая возможные на данном томе типы атрибутов файлов;
- *$Root Directory* – файл с информацией о корневом каталоге тома. В нем хранятся ссылки на файлы и каталоги, содержащиеся в корневом каталоге;
- *$BitMap* – файл битовой карты, каждый бит в которой соответствует одному кластеру: единичное значение бита соответствует занятому кластеру, нулевое – свободному;
- *$Boot* – файл загрузочной записи тома;
- *$BadClus* – файл плохих кластеров, содержащий информацию обо всех кластерах, имеющих сбойные секторы.

Альтернативные потоки NTFS

Файловая система *NTFS* обладает интересной возможностью поддержки альтернативных потоков данных (*Alternate Data Stream, ADS*). Технология подразумевает под собой, то, что каждый файл в файловой системе *NTFS* может иметь несколько потоков, в которых могут храниться данные. Проводник и большинство других приложений работают только со стандартным потоком и не могут получить данные их альтернативных. Таким образом с помощью технологии *ADS* можно скрывать данные, которые не удастся обнаружить стандартными способами. Поддержка альтернативных потоков данных была добавлена в *NTFS* для совместимости с файловой системой *HFS*, использующейся на *MacOS*.

Потоки данных файла описываются атрибутом *$DATA*, одним из свойств которого является имя потока. Основной поток является неименованным, а каждый альтернативный поток должен иметь собственное имя. Альтернативные потоки скрыты от пользователя и не отображаются большинством стандартных программ. Они могут содержать любой тип информации – текстовый, графический, видео и т.д. В альтернативный поток можно даже записать программу, что иногда используется для распространения вредоносного ПО.

Пример.

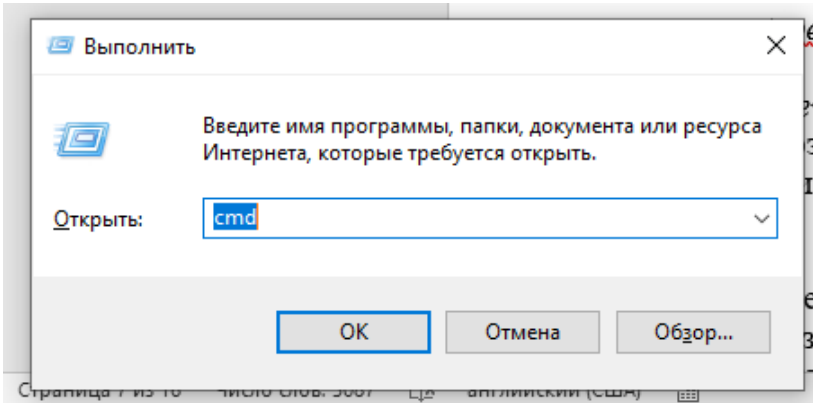
1. Создаем в *Блокноте (Notepad)* файл ***primer.txt*** размером 15 байтов, содержащий строку*:"Hello, students!"*

2. Создаем в этом файле именованный поток с именем ***potok1.txt***:

echo This is second stream > primer.txt:potok1.txt

Для выполнения этой и следующих команд необходимо:

- Запустить командную строку с использованием диалога ***Выполнить*** (комбинации клавиш ***Win+R***).
- Набрать интерпретатор командной строки ***cmd***.
- *OK.*
- Вставить скопированную команду в командную строку.
- *Enter.*



Размер файла, выводимый командой ***dir*** или Проводником, при этом не изменится.

3. Просмотрим содержимое основного и альтернативного потоков:

type primer.txt (выводится строка «Hello,students!»)

type primer.txt:potok1.txt (ошибка, команда не видит второй поток !)

more< primer.txt:potok1.txt (выводится строка «This is second stream»)

4. Создадим в файле второй альтернативный поток с именем ***potok2.jpg***, содержащий графический файл:

type potok2.jpg > primer.txt:potok2.jpg

Размер файла, выводимый командой ***dir*** или Проводником, при этом также не изменится.

5. Извлечем графические данные из потока с помощью графического редактора:

mspaint primer.txt:potok2.jpg

6. Создадим в файле третий альтернативный поток с именем ***calcul.exe***, содержащий программу Калькулятор:

type c:\windows\system32\ calc.exe > primer.txt:calcul.exe

Размер файла, выводимый командой ***dir*** или Проводником, при этом опять не изменится.

7. Запустим Калькулятор из текстового файла:

start .\ primer.txt:calcul.exe

Действительный размер полученных в этом примере файлов можно увидеть командой ***dir/r.***

Программное обеспечение для выполнения работы

Основным инструментом исследования файловой системы магнитных дисков является специальная программа – дисковый редактор. Современные дисковые редакторы обладают большим набором возможностей: просмотр и редактирование системной области диска; просмотр и редактирование директорий и файлов; восстановление удаленных директорий и файлов; доступ к любому участку диска по номеру сектора или кластера; работа с образом диска; создание загрузочных дисков и т.д.

Для выполнении лабораторной работы будем использовать редактор *DMDE* (*http://dmde.ru*). Редактор *DMDE* имеет свободно распространяемую версию, которую можно скачать с сайта разработчика, поэтому рекомендуем работать именно с этой программой. Выполнение данной лабораторной работы требует прямого обращения к дискам, что несет потенциальную опасность для вычислительной системы, поэтому для работы необходимо использовать дисковый редактор ***только в режиме чтения***.

Если Вы выполняете работу на домашнем компьютере, то можно работать с реальными дисками или флэш – накопителями. В компьютерном классе в целях безопасности администратор системы может отключить возможность работы с дисками, в этом случае можно работать с заранее подготовленным образом диска или использовать виртуальную машину, работающую в любой среде виртуализации (*Oracle VirtualBox*, *VM Workstation* и т.д.)

После загрузки редактора необходимо выбрать тип диска – физический или логический. При выборе физического диска открывается таблица разделов, в которой хранится список логических дисков с указанием типа файловой системы, объема и границ каждого логического диска (рис. 4). Для отображения имени разделов диска можно нажать кнопку «Меню» и выбрать пункт «Показать буквы томов».

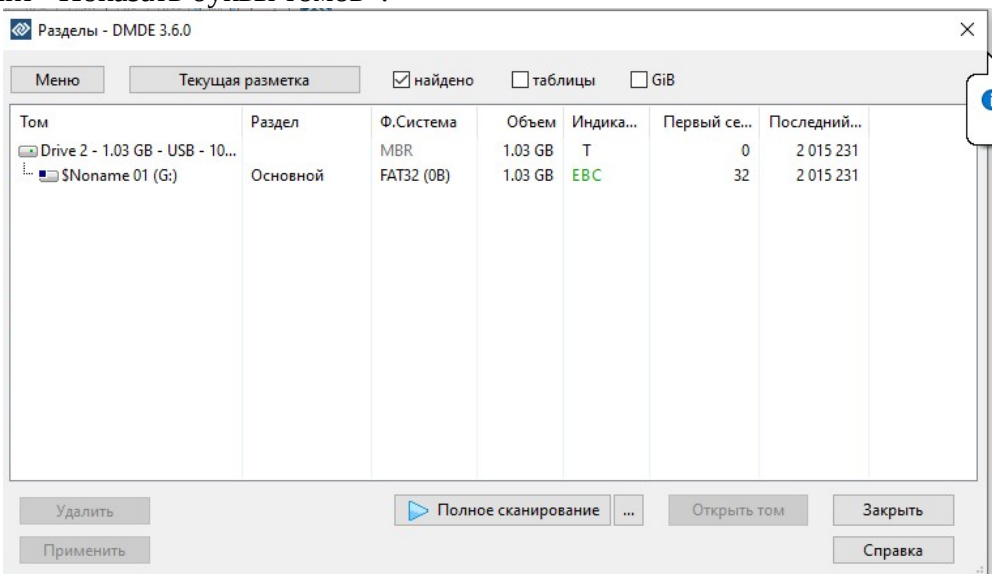


Рис. 4

С помощью контекстного меню для каждого логического диска можно выполнить следующие действия: открыть, удалить или создать образ. Образ представляет собой файл, содержащий снимок диска, т.е. его точную физическую копию, которую можно использовать для восстановления диска в случае повреждения.

После открытия логического диска редактор выводит его параметры, набор которых зависит от типа установленной файловой системы: размеры сектора и кластера, число элементов корневого каталога или расположение файла *MFT* и т.д. (рис. 5).

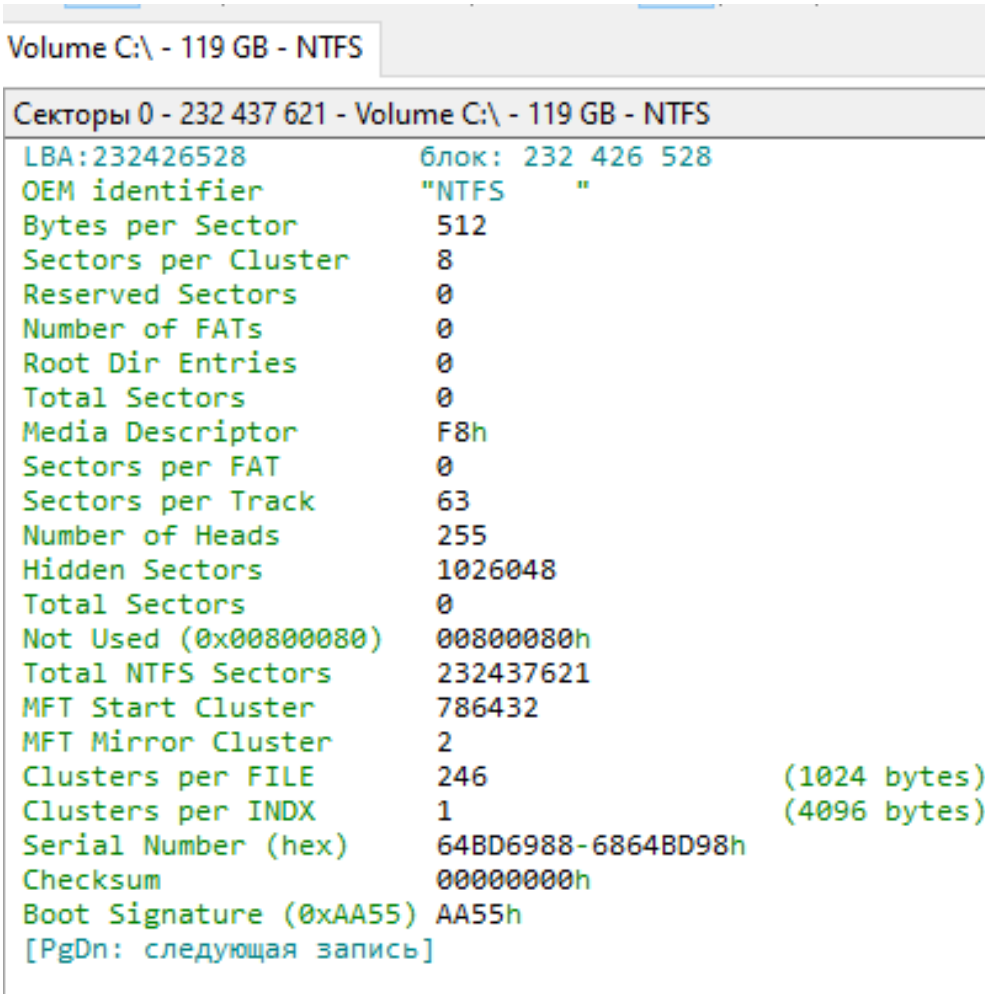


Рис. 5

Нажатие кнопки «Открыть» переводит редактор в режим просмотра, в котором имеется три панели (просмотр папок, просмотр файлов и панель редактора), показанных на рис. 6. В панель редактора можно выводить содержимое системной области и области данных диска. Управление панелью редактора проводится через меню Редактор.

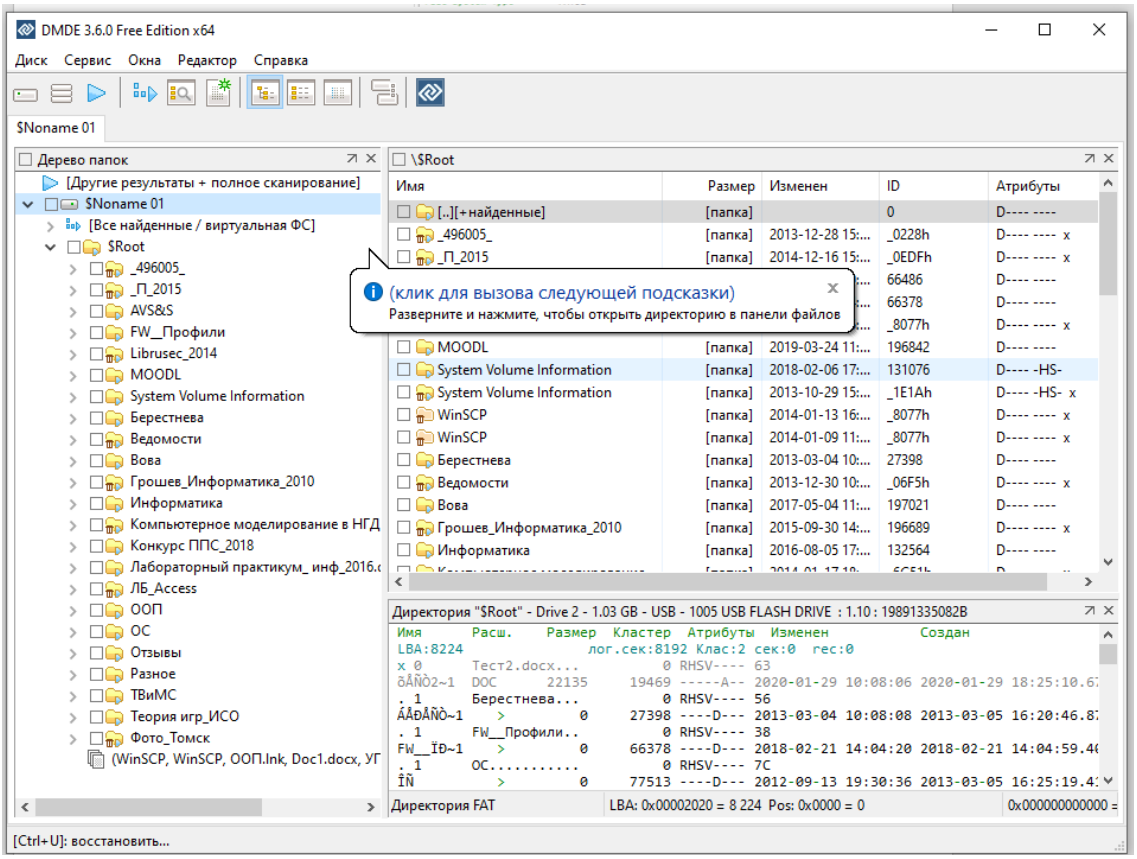


Рис.6

Работа с файловой системой FAT32

Для *FAT*32 в панель редактора из системной области можно выводить загрузочную запись, таблицу *FAT* и корневой каталог, а из области данных – каталоги и файлы.

При просмотре таблицы *FAT* элементы, соответствующие свободным кластерам, выводятся символом «0», занятым кластерам – символом «=», а занятым последним кластерам – символом «*E*». Реальные значения элементов *FAT* выводятся при установке курсора на элемент, при этом в строке статуса отображается название файловой системы и номер кластера, который соответствует текущему элементу *FAT* (например, *FAT*32 [17251]). По каждому каталогу и файлу выводится имя, расширение, размер, номер начального кластера, атрибуты и даты создания и изменения (см. рис. 21).

Просмотр содержимого файла, которое выводится в шестнадцатиричном и символьном виде, проводится двойным щелчком мыши по имени файла; изменение кодировки символов проводится в меню Режим/Кодировка. В этом режиме можно также посмотреть цепочку кластеров, выделенных данному файлу (меню Редактор/ Карта кластеров), как показано на рис.7.

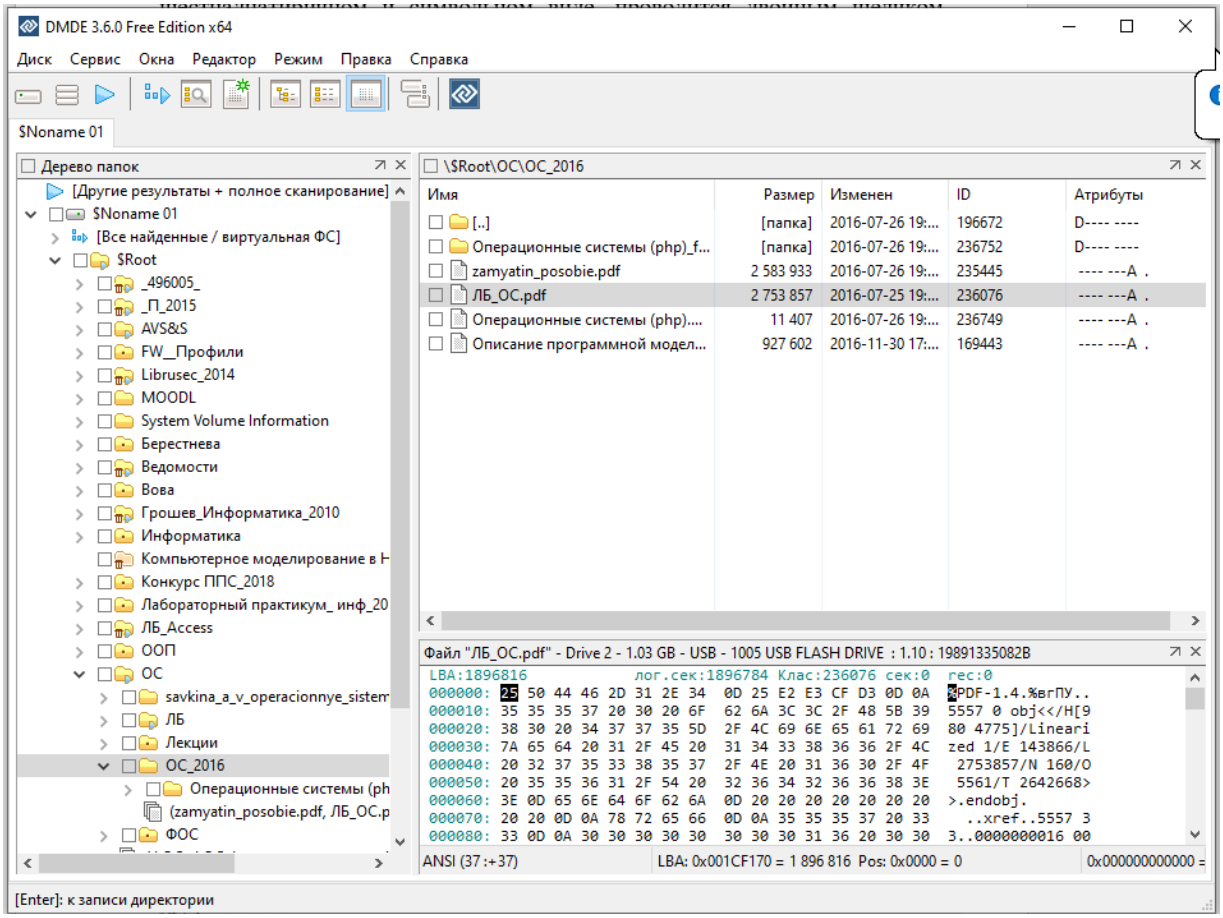


Рис.7

Элементы каталога, имена которых начинаются с символа «*x*» соответствуют удаленным файлам. Если в поле имени стоят цифры или символы «e0», то этот элемент предназначен для хранения длинного имени файла.

DMDE позволяет осуществить быстрый переход на заданный кластер или сектор диска по их номеру (меню Редактор/Кластер или меню Редактор/Сектор тома), а также восстановить удаленные файлы.

Для восстановления необходимо отметить на панели нужные файлы, выбрать в контекстном меню пункт «Восстановить объект…» и указать каталог, в который надо провести восстановление. Для того, чтобы не испортить файл-оригинал, восстановление желательно проводить на другой логический диск.

Работа с файловой системой NTFS

После выбора логического диска в окне редактора будет выведено содержимое файла *$MFT* (рис. 8).

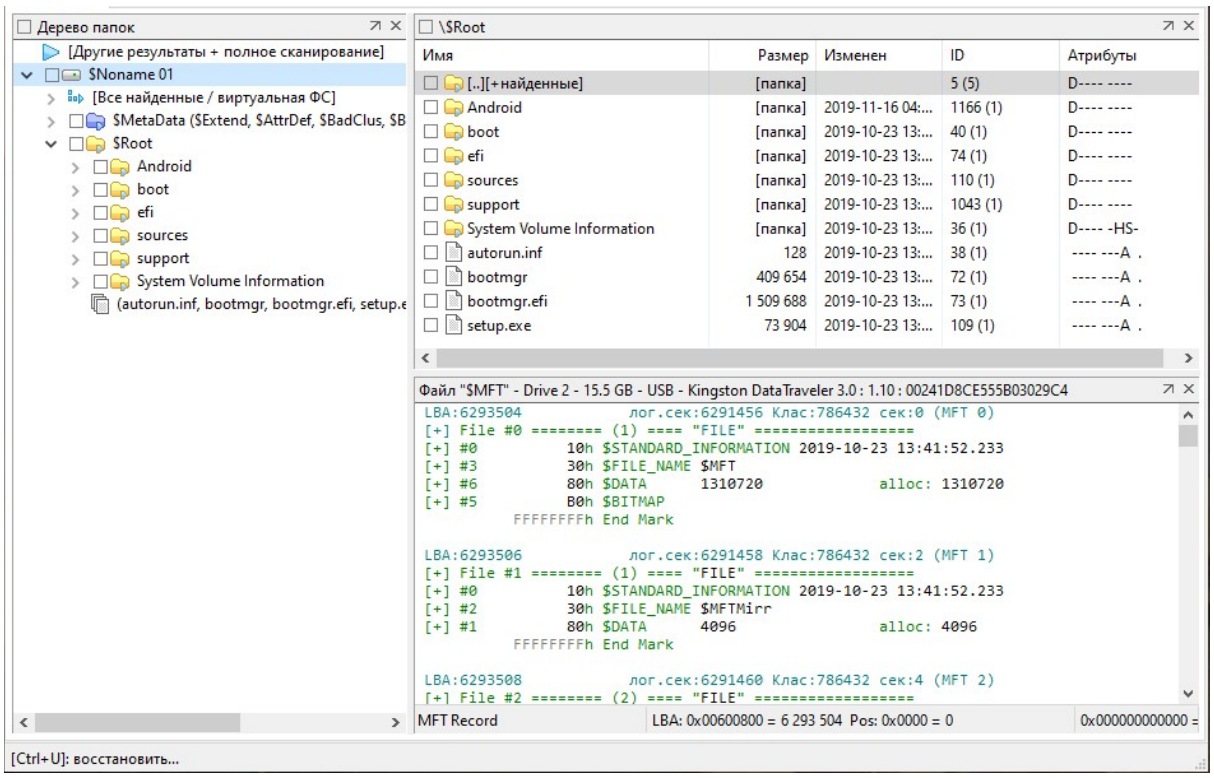


Рис. 8

Первая запись описывает сам файл *$MFT*, а вторая – копию его первых четырех записей (*$MFTMirr*). Для каждой записи выводится ее адрес на диске (номера кластера и сектора), граничные метки, внутренний номер (индекс) и набор атрибутов. Минимальный набор включает атрибуты *$STAN-DART INFORMATION*, *$FILE NAME* и *$DATA*. Для просмотра содержимого каждого атрибута необходимо в его строке сделать щелчок мыши на символе ‘+’.

На рис. 9 показано содержимое атрибута *$DATA*, указывающего на расположение данных одного из файлов.

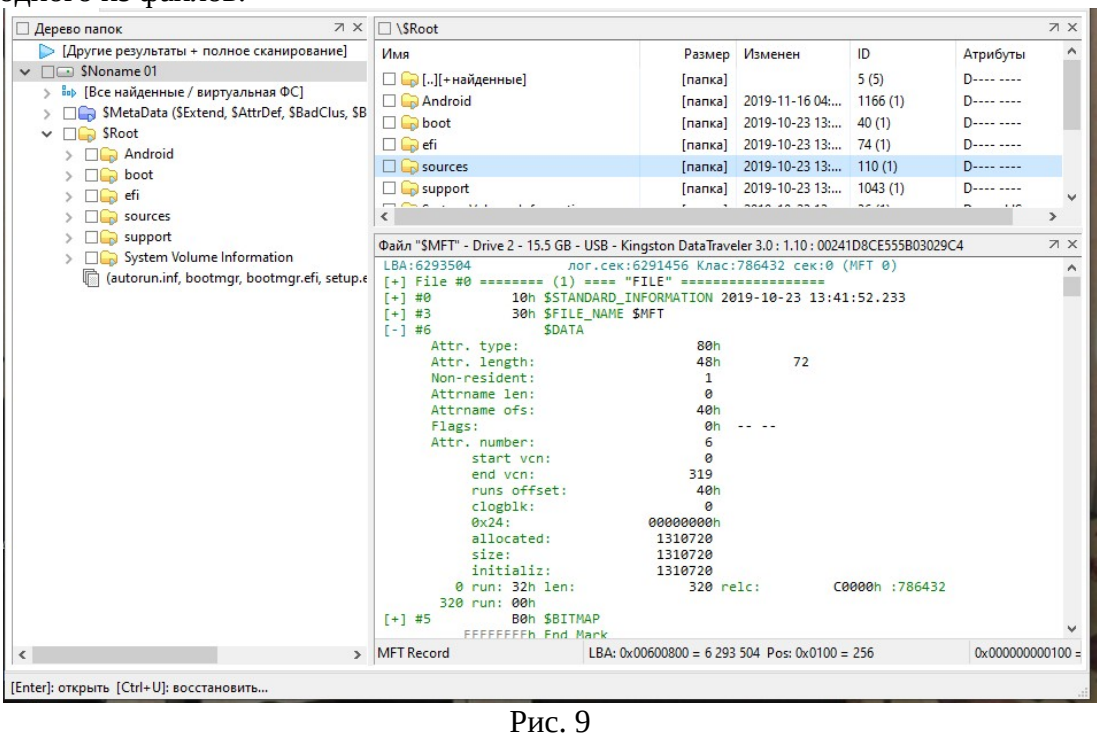


Рис. 9

Анализ рисунка позволяет сделать следующие выводы:

- индекс файла в MFT – 110;
- данные файла находятся на диске, т.к. атрибут является нерезидентным;

- данные занимают 320 кластеров (start vcn=0, end vcn=319) или 320*8*512 = 1310720 байтов, файл не фрагментирован;
- номер начального кластера файла – 786432;
- длина атрибута – 72 байта.

На рис.10 показан этот файл в режиме просмотра данных

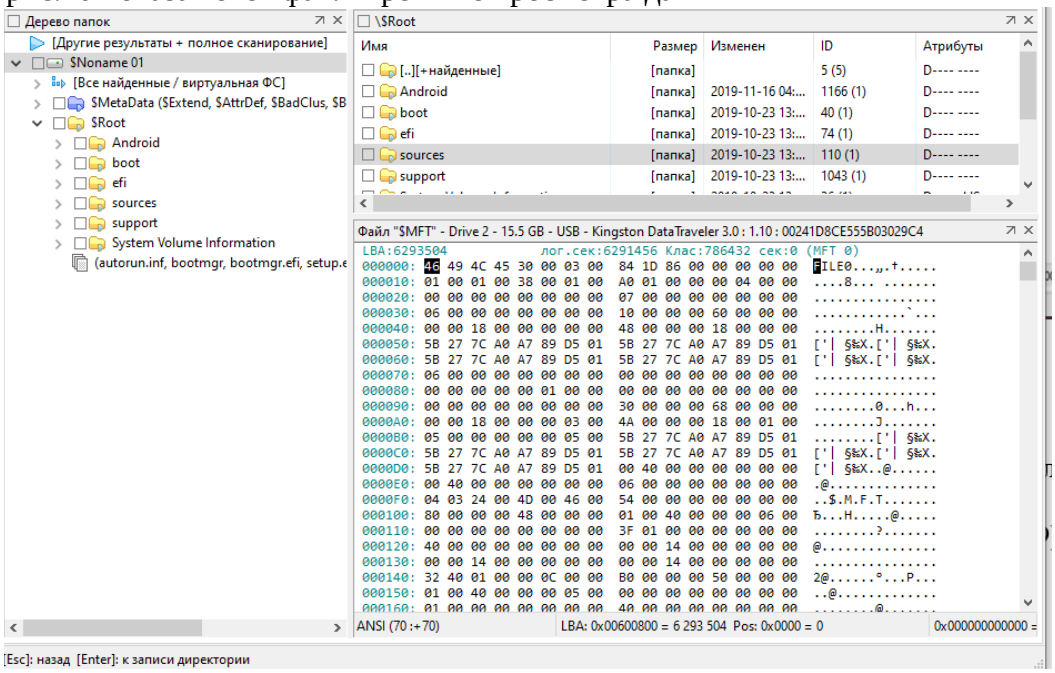


Рис. 10

Порядок выполнения работы

1. Войдите в среду Windows на рабочем компьютере с помощью бригадной учетной записи и подключитесь к виртуальной машине.

2. Откройте дисковый редактор DMDE и определите параметры виртуального диска: общий объем, число и типы разделов, тип файловой установленной файловой системы. Для FAT - раздела определите размеры сектора и кластера; число секторов, выделенных для таблицы FAT и размер корневого каталога. Для NTFS - раздела определите размеры сектора и кластера, размер файла $MFT и его адрес, размеры записи MFT и индексной записи. Занесите все параметры в отчет, подтверждая их скриншотами.

3. Откройте логический диск с файловой системой FAT32 и выполните следующие действия, подтверждая их скриншотами:

3.1 Создайте на диске каталог с именем, соответствующим Вашей учетной записи, и в нем создайте структуру каталогов согласно заданию лабораторной работы № 1.

3.2. В каталог *abc_kk* запишите три файла размером 40 – 60 Кбайт, имеющих форматы *.txt*, *.doc* и *.docx* , имена файлов должны содержать не менее 15 символов, например, *Лабораторная работа № 3*.

3.3. С помощью программы *Проводник* скопируйте файл *Лабораторная работа № 3.txt* в каталог *trash_kk***.**

3.4. Удалите файл *Лабораторная работа № 3.txt* из каталога *abc_kk*, проведите анализ изменений в FAT и в каталоге *abc_kk*. Посмотрите содержимое начального кластера удаленного файла, результат занесите в отчет.

3.5. Восстановите удаленный файл *Лабораторная работа № 3.txt.*

4. Откройте логический диск с файловой системой NTFS и выполните действия, подтверждая их скриншотами.

4.1. Создайте на диске структуру каталогов и файлов согласно п.3.1 и п.3.2.

4.2. Определите характеристики файла $MFT (начальный адрес, число записей, размер в байтах и кластерах).

4.3. Определите число записей в файле $MFTmirr.

4.4. Проведите полный анализ записи MFT, соответствующей файлу *Лабораторная работа № 3.txt* и занесите в отчет описания всех атрибутов, включая расположение файла на диске.
4.5. Удалите файл *Лабораторная работа №3.txt*, проведите анализ изменений в MFT и в области данных. Результаты занесите в отчет.
4.6. Восстановите удаленный файл.
4.7. С помощью программы Блокнот создайте текстовый файл **primer.txt**, записав в него фразу «Very good weather today!». Проведите анализ соответствующей записи MFT, определить адрес этого файла на диске.
4.8. Запишите в файл **primer.txt** второй поток данных, используя для этого, например, любой текстовый файл размером не менее 50 Кбайт. Проведите анализ соответствующей записи MFT и определите расположение данных этого потока на диске. Определите размер файла, сравните с предыдущим пунктом.
4.9. Запишите в файл **primer.txt** третий поток данных, используя для этого любой графический файл (например, фотографию). Проведите анализ соответствующей записи MFT и определите расположение данных этого потока на диске. Определите размер файла, сравните с предыдущим пунктом.

Контрольные вопросы

1. Каким образом поддерживается древовидная многоуровневая система каталогов в Windows?
4. Какова структура FAT, в чем отличия для жестких и гибких дисков?
5. Какова структура каталогов файловой системы FAT? В чем отличие корневого и прочих каталогов?
6. Какие действия выполняются файловой системой при удалении файла в файловых системах FAT и NTFS?
7. Поясните действия файловой системы FAT при поиске файла по имени:
а) файл находится в корневом каталоге;
б) файл расположен в подкаталоге.
8. Поясните механизм выделения дисковой памяти файловой системы FAT при записи нового файла на диск.
9. Какие компоненты компьютера используют физическую и логическую модели магнитного диска?
10. Чем определяется число элементов каталога, выделяемых для хранения метаданных файла в файловой системе FAT?
11. Назовите основные различия файловых систем FAT и NTFS.
12. Какова структура файла MFT?
13. Поясните структуру файловой записи MFT.
14. Алгоритмы восстановления файлов в FAT и NTFS.
15. Резидентные и нерезидентные атрибуты записи MFT.
16. Каким образом в NTFS увеличена скорость доступа к файлам по сравнению с FAT?

Практическая работа №3

Изучение структуры операционной системы

Цель занятия: изучить структуру операционной системы, назначение ее отдельных компонентов, функции операционных систем, их классификацию и критерии оценки.

Задание: составить конспект основных теоретических положений.

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Операционная система (operating system) – комплекс программ, предоставляющий пользователю удобную среду для работы с компьютерным оборудованием.

Операционная система позволяет запускать пользовательские программы; управляет всеми ресурсами компьютерной системы – процессором (процессорами), оперативной памятью,

устройствами ввода вывода; обеспечивает долговременное хранение данных в виде файлов на устройствах внешней памяти; предоставляет доступ к компьютерным сетям.

Для более полного понимания роли операционной системы рассмотрим составные компоненты любой вычислительной системы (рис.1.1).

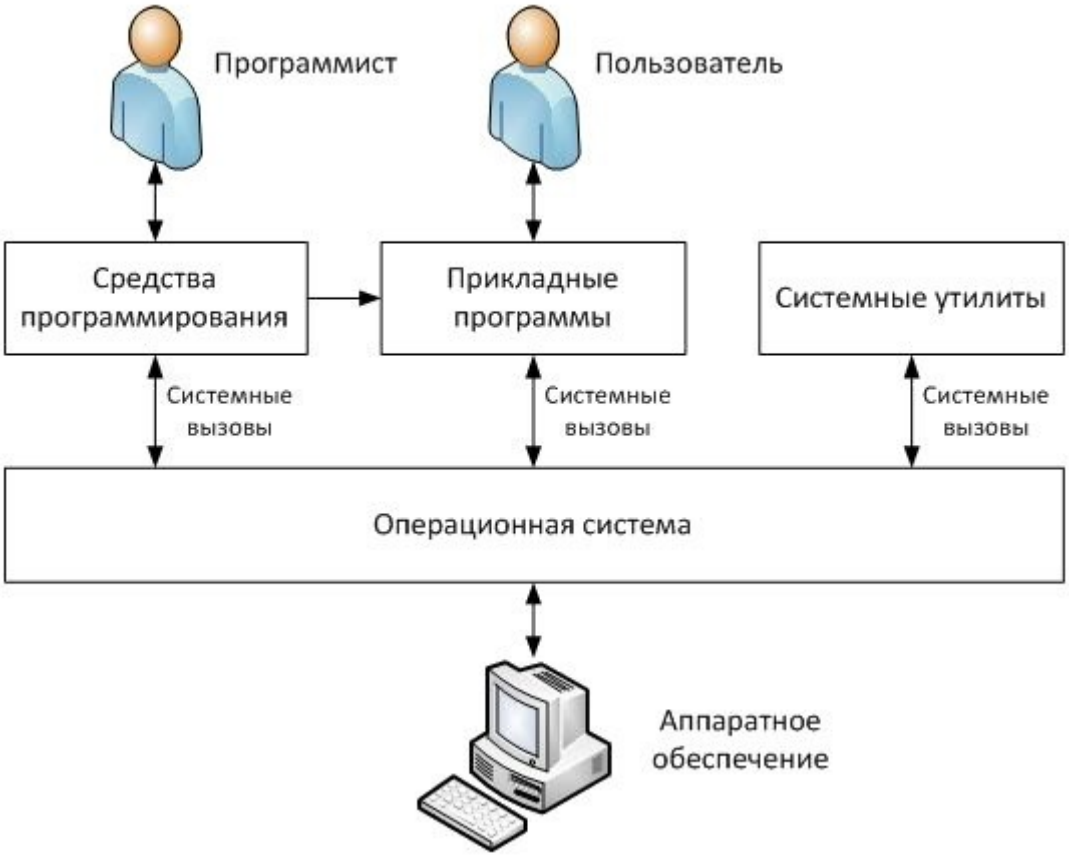


Рис. 1.1. Компоненты вычислительной системы

Все компоненты можно разделить на два больших класса – программы или программное обеспечение (ПО, software) и оборудование или аппаратное обеспечение (hardware). Программное обеспечение делится на прикладное, инструментальное и системное.

Цель создания вычислительной системы – решение задач пользователя. Для решения определенного круга задач создается прикладная программа (приложение, application). Примерами прикладных программ являются текстовые редакторы и процессоры (Блокнот, Microsoft Word), графические редакторы (Paint, Microsoft Visio), электронные таблицы (Microsoft Excel), системы управления базами данных (Microsoft Access, Microsoft SQL Server), браузеры (Internet Explorer) и т. п. Все множество прикладных программ называется прикладным программным обеспечением (application software).

Создается программное обеспечение при помощи разнообразных средств программирования (среды разработки, компиляторы, отладчики и т. д.), совокупность которых называется инструментальным программным обеспечением. Представителем инструментального ПО является среда разработки Microsoft Visual Studio.

Основным видом системного программного обеспечения являются операционные системы. Их основная задача – обеспечить интерфейс (способ взаимодействия) между пользователем и приложениями с одной стороны, и аппаратным обеспечением с другой. К системному ПО относятся также системные утилиты – программы, которые выполняют строго определенную функцию по обслуживанию вычислительной системы, например, диагностируют состояние системы, выполняют дефрагментацию файлов на диске, осуществляют сжатие (архивирование) данных. Утилиты могут входить в состав операционной системы.

Взаимодействие всех программ с операционной системой осуществляется при помощи системных вызовов (system calls) – запросов программ на выполнение операционной системой необходимых действий. Набор системных вызовов образует API – Application Programming Interface (интерфейс прикладного программирования).

Функции операционной системы

К основным функциям, выполняемым операционными системами, можно отнести:

- обеспечение выполнения программ – загрузка программ в память, предоставление программам процессорного времени, обработка системных вызовов;
- управление оперативной памятью – эффективное выделение памяти программам, учет свободной и занятой памяти;
- управление внешней памятью – поддержка различных файловых систем;
- управление вводом-выводом – обеспечение работы с различными периферийными устройствами;
- предоставление пользовательского интерфейса;
- обеспечение безопасности – защита информации и других ресурсов системы от несанкционированного использования;
- организация сетевого взаимодействия.

Структура операционной системы

Современные процессоры имеют минимум два режима работы – привилегированный (supervisor mode) и пользовательский (user mode).

Отличие между ними заключается в том, что в пользовательском режиме недоступны команды процессора, связанные с управлением аппаратным обеспечением, защитой оперативной памяти, переключением режимов работы процессора. В привилегированном режиме процессор может выполнять все возможные команды.

Приложения, выполняемые в пользовательском режиме, не могут напрямую обращаться к адресным пространствам друг друга – только посредством системных вызовов.

Все компоненты операционной системы можно разделить на две группы – работающие в привилегированном режиме и работающие в пользовательском режиме, причем состав этих групп меняется от системы к системе.

Основным компонентом операционной системы является ядро (kernel). Функции ядра могут существенно отличаться в разных системах; но во всех системах ядро работает в привилегированном режиме (который часто называется режим ядра, kernel mode).

Ядро - это основная, «самая системная» часть операционной системы. Имеются разные определения ядра. Согласно одному из них, ядро — это ***резидентная*** часть системы, т.е. к ядру относится тот программный код, который постоянно находится в памяти в течение всей работы системы. Остальные модули ОС являются ***транзитными***, т.е. подгружаются в память с диска по мере необходимости на время своей работы. К транзитным частям системы относятся:

утилиты (utilities) — отдельные системные программы, решающие частные задачи, такие как форматирование и проверку диска, поиск данных в файлах, мониторинг (отслеживание) работы системы и многое другое;

системные библиотеки подпрограмм, позволяющие прикладным программам использовать различные специальные возможности, поддерживаемые системой (например, библиотеки для графического вывода, для работы с мультимедиа и т.п.);

интерпретатор команд — программа, выполняющая ввод команд пользователя, их анализ и вызов других модулей для выполнения команд;

системный загрузчик — программа, которая при запуске ОС (например, при включении питания) обеспечивает загрузку системы с диска, ее инициализацию и старт;

другие виды программ, в зависимости от конкретной системы.

Особую роль в структуре системы играют ***драйверы устройств***. Эти программы, предназначенные для обслуживания конкретных периферийных устройств, несомненно, можно отнести к ядру системы: они почти всегда являются резидентными и работают в режиме ядра. Но в отличие от самого ядра, которое изменяется только при появлении новой версии ОС,

набор используемых драйверов весьма мобилен и зависит от набора устройств, подключенных к данному компьютеру. В некоторых системах (например, в ранних версиях UNIX) для подключения нового драйвера требовалось перекомпилировать все ядро. В большинстве современных ОС драйверы подключаются к ядру в процессе загрузки системы, а иногда разрешается даже загрузка и выгрузка драйверов в ходе работы системы.

В качестве программного интерфейса системы, т.е. средств для обращения прикладных программ к услугам ОС, используется документированный набор ***системных вызовов*** или ***функций API*** (Applied Programming Interface). Между этими двумя терминами есть некоторая разница. Под системными вызовами понимаются функции, реализуемые непосредственно программами ядра системы. При их выполнении происходит переход из режима пользователя в режим ядра, а затем обратно. В отличие от этого, API-функции определяются как функции, описанные в документации ОС, независимо от того, выполняются ли они ядром или же системными библиотеками, работающими в режиме пользователя. В Windows часто несколько разных API-функций обращаются к одному и тому же недокументированному системному вызову, но имеют различные обрамляющие части, работающие в режиме пользователя.

Там, где различие между двумя этими понятиями несущественно, можно использовать нейтральный термин «***системные функции***».

Существует два основных вида ядер – монолитные ядра (monolithic kernel) и микроядра (microkernel). В монолитном ядре реализуются все основные функции операционной системы, и оно является, по сути, единой программой, представляющей собой совокупность процедур. В микроядре остается лишь минимум функций, который должен быть реализован в привилегированном режиме: планирование потоков, обработка прерываний, межпроцессное взаимодействие. Остальные функции операционной системы по управлению приложениями, памятью, безопасностью и пр. реализуются в виде отдельных модулей в пользовательском режиме.

Ядра, которые занимают промежуточные положение между монолитными и микроядрами, называют гибридными (hybrid kernel).

Примеры различных типов ядер:

- монолитное ядро – MS-DOS, Linux, FreeBSD;
- микроядро – Mach, Symbian, MINIX 3;
- гибридное ядро – NetWare, BeOS, Syllable.

Кроме ядра в привилегированном режиме (в большинстве операционных систем) работают драйверы (driver) – программные модули, управляющие устройствами.

В состав операционной системы также входят:

- системные библиотеки (system DLL – Dynamic Link Library, динамически подключаемая библиотека), преобразующие системные вызовы приложений в системные вызовы ядра;
- пользовательские оболочки (shell), предоставляющие пользователю интерфейс – удобный способ работы с операционной системой.

Пользовательские оболочки реализуют один из двух основных видов пользовательского интерфейса:

- текстовый интерфейс (Text User Interface, TUI), другие названия – консольный интерфейс (Console User Interface, CUI), интерфейс командной строки (Command Line Interface, CLI);
- графический интерфейс (Graphic User Interface, GUI).

Пример реализации текстового интерфейса в Windows – интерпретатор командной строки cmd.exe; пример графического интерфейса – Проводник Windows (explorer.exe).

Классификация операционных систем

Классификацию операционных систем можно осуществлять несколькими способами.

1. По способу организации вычислений:

- системы пакетной обработки (batch processing operating systems) – целью является выполнение максимального количества вычислительных задач за единицу времени; при этом из нескольких задач формируется пакет, который обрабатывается системой;
- системы разделения времени (time-sharing operating systems) – целью является возможность одновременного использования одного компьютера несколькими пользователями; реализуется посредством поочередного предоставления каждому пользователю интервала процессорного времени;
- системы реального времени (real-time operating systems) – целью является выполнение каждой задачи за строго определённый для данной задачи интервал времени.

2. По типу ядра:
- системы с монолитным ядром (monolithic operating systems);
- системы с микроядром (microkernel operating systems);
- системы с гибридным ядром (hybrid operating systems).

3. По количеству одновременно решаемых задач:
- однозадачные (single-tasking operating systems);
- многозадачные (multitasking operating systems).

4. По количеству одновременно работающих пользователей:
- однопользовательские (single-user operating systems);
- многопользовательские (multi-user operating systems).

5. По количеству поддерживаемых процессоров:
- однопроцессорные (uniprocessor operating systems);
- многопроцессорные (multiprocessor operating systems).

6. По поддержке сети:
- локальные (local operating systems) – автономные системы, не предназначенные для работы в компьютерной сети;
- сетевые (network operating systems) – системы, имеющие компоненты, позволяющие работать с компьютерными сетями.

7. По роли в сетевом взаимодействии:
- серверные (server operating systems) – операционные системы, предоставляющие доступ к ресурсам сети и управляющие сетевой инфраструктурой;
- клиентские (client operating systems) – операционные системы, которые могут получать доступ к ресурсам сети.

8. По типу лицензии:
- открытые (open-source operating systems) – операционные системы с открытым исходным кодом, доступным для изучения и изменения;
- проприетарные (proprietary operating systems) – операционные системы, которые имеют конкретного правообладателя; обычно поставляются с закрытым исходным кодом.

9. По области применения:
- операционные системы мэйнфреймов – больших компьютеров (mainframe operating systems);
- операционные системы серверов (server operating systems);
- операционные системы персональных компьютеров (personal computer operating systems);
- операционные системы мобильных устройств (mobile operating systems);
- встроенные операционные системы (embedded operating systems);
- операционные системы маршрутизаторов (router operating systems).

Требования к операционным системам

Основное требование, предъявляемое к современным операционным системам – выполнение функций, перечисленных выше в параграфе "Функции операционных систем". Кроме этого очевидного требования существуют другие, часто не менее важные:

- расширяемость – возможность приобретения системой новых функций в процессе эволюции; часто реализуется за счет добавления новых модулей;
- переносимость – возможность переноса операционной системы на другую аппаратную платформу с минимальными изменениями;

- совместимость – способность совместной работы; может иметь место совместимость новой версии операционной системы с приложениями, написанными для старой версии, или совместимость разных операционных систем в том смысле, что приложения для одной из этих систем можно запускать на другой и наоборот;
- надежность – вероятность безотказной работы системы;
- производительность – способность обеспечивать приемлемые время решения задач и время реакции системы.

Предыстория ОС

Вскоре после того, как в конце 40-х годов XX века были созданы первые электронные компьютеры, очень остро встала проблема повышения эффективности использования оборудования, и прежде всего центрального процессора.

Типичный компьютер первого — второго поколений представлял собой большую комнату, уставленную шкафами и увитую кабелями. Каждое из основных устройств — центральный процессор, оперативная память, накопители на магнитных лентах, устройства ввода с перфокарт, принтер — занимало один или несколько «шкафов» или «тумб», наполненных радиолампами и механическими частями.

Все это стоило больших денег, потребляло бешеное количество электроэнергии и регулярно ломалось.

В таких условиях машинное время стоило очень дорого. Тем не менее, обычная практика использования ЭВМ не способствовала экономии. Как правило, программист, разрабатывающий программу, заказывал ежедневно несколько часов машинного времени и в течение этого времени монопольно использовал машину. Выполнив очередной запуск отлаживаемой программы (которую надо было каждый раз вводить либо с клавиатуры, либо, в лучшем случае, с перфокарт), пользователь получал распечатку (чаще всего в виде массива цифр), анализировал результаты, вносил изменения в программу и снова запускал ее. Таким образом, в ходе сеанса отладки дорогостоящее оборудование простаивало 99% времени, пока программист осмысливал результаты и работал с устройствами ввода/вывода. Кроме того, сбой при вводе одной перфокарты мог потребовать начать сначала всю работу программы.

Возникла великая идея — использовать сам компьютер для повышения эффективности работы с ним же.

Одно из ответвлений этой идеи — создание языков и систем программирования — рассматривается в отдельных курсах. Другим важным шагом стало возложение на специальную компьютерную программу части тех функций, которые до этого выполнял оператор или сам программист.

Программы такого рода назывались обычно ***мониторами*** (не путать с монитором как устройством вывода, который в то время был редчайшей экзотикой!). Монитор принимал команды, состоящие, как правило, из 1-2 букв названия и 1-3 аргументов, заданных 8-ричными или 16-ричными числами. Типичными командами были, например:

загрузка данных с перфокарт по указанному адресу памяти;

просмотр и корректировка (с пишущей машинки) значений в указанном диапазоне адресов;

пошаговое выполнение программы с выдачей результатов каждой команды на пишущую машинку;

запуск программы с указанного адреса с заданием адресов контрольных точек остановки.

Несмотря на убогость, по нынешним меркам, подобных средств, они в свое время значительно повысили производительность работы программистов. Однако кардинального повышения загрузки процессора не произошло.

Временем широкого распространения мониторов в мире были 50-е годы прошлого века (в СССР — 60-е годы). В настоящее время нечто подобное можно встретить на самых примитивных микропроцессорных контроллерах.

Пакетные ОС

Историю собственно ОС можно начать с появления в конце 50-х годов первых систем, организующих работу по пакетному принципу.

Важнейшим организационным изменением, происшедшим на этом этапе развития, стало массовое изгнание программистов из машинных залов, как фактора, лишь вносящего сумятицу в работу.

Теперь от программиста требовалось собрать пакет перфокарт, содержащий его программу, данные к ней, а также управляющие перфокарты. Эти карты на специально разработанном ***языке управления заданиями*** (JCL, Job Control Language) объясняли операционной системе, чье это задание, что нужно сделать с программой (например, передать ее транслятору с Фортрана), что предпринять в случае успешной трансляции (вероятно, пустить на решение), что — при наличии ошибок (например, перейти к другой программе), откуда взять исходные данные (например, с такого-то цилиндра магнитного диска). Кроме того, там могли быть даже указания на то, сколько метров бумаги можно выделить на распечатку и какое максимальное время может занять работа программы.

Обойтись без столь подробных инструкций было нельзя, потому что программист не присутствовал при запуске задания и не мог вмешаться лично.

Подготовленный пакет передавался, вместе с другими подобными пакетами, оператору ЭВМ, перед которым стояли две основные задачи: чтобы в устройстве ввода не переводились пакеты заданий и чтобы в принтере не кончилась бумага. Когда процессор заканчивал обработку задания и печать его результатов, он вводил следующий пакет и приступал к его обработке. Так достигалась основная цель пакетного режима — исключить простои процессора из-за нерасторопности людей.

В скором времени разработчики ОС осознали, что вычерпаны далеко не все резервы повышения загрузки процессора. Операции ввода и печати требовали лишь очень небольшой доли от полной производительности процессора. Кроме того, в ходе работы программы случались обращения к периферийным устройствам (например, к магнитным лентам и, позднее, дискам), при выполнении которых процессор опять простаивал. Целесообразно было найти способ, чтобы в эти периоды ожидания загрузить процессор другой работой. Но для этого необходимо, чтобы в памяти процессора находились сразу несколько программ, тогда ОС смогла бы переключать процессор на выполнение той программы, которая в данный момент может работать.

Такая организация работы, когда в памяти находятся несколько программ и система в определенные моменты переключает выполнение с одной программы на другую, была названа ***мультипрограммированием***. Эта важная идея в разных воплощениях пережила те пакетные системы, в которых она впервые была реализована, и является основой для функционирования практически всех современных ОС.

Среди наиболее развитых пакетных ОС с мультипрограммированием нельзя не назвать OS/360, основную ОС знаменитого в 60-70 гг. семейства ЭВМ IBM 360/370.

ОС с разделением времени

На рубеже 60-70 гг. распространенным и не слишком дорогим периферийным устройством становятся мониторы (сначала монохромные и работающие только в текстовом режиме). При этом процессор и ОЗУ остаются самыми дорогими и громоздкими устройствами вычислительной системы. В этих условиях возникает и быстро приобретает популярность принципиально новый тип ОС — ***системы с разделением времени***.

К одной ЭВМ подключается несколько десятков рабочих мест, оборудованных дисплеем (монитор + клавиатура) и совместно использующих вычислительные ресурсы ЭВМ. Процессорное время делится на кванты длительностью в несколько десятков миллисекунд и по истечении каждого кванта процессор может быть переключен на обслуживание другого процесса, другого дисплея. Поскольку теперь подготовку текстов программ выполняют сами программисты за дисплеями, а работа по редактированию текста требует очень малых затрат процессорного времени, процессор успевает обслужить все рабочие места практически без

ощутимой задержки. Большая часть времени процессора уделяется небольшому числу рабочих мест, где в данный момент запущены на выполнение программы. При этом, разумеется, средняя скорость работы каждой программы уменьшается, по крайней мере во столько раз, сколько программ выполняется одновременно.

Режим разделения времени стал огромным облегчением для программистов, которые вновь смогли в некоторой степени почувствовать себя «хозяевами» ЭВМ и получили возможность запускать программы на трансляцию и отладку хоть каждые 5 минут. Это позволило сократить сроки разработки и отладки программ.

Для трудоемких вычислительных заданий, предусматривающих счет по ранее отлаженным программам, режим разделения времени менее эффективен, чем пакетный, поскольку частое переключение процессора между выполняемыми программами требует дополнительных затрат времени.

Системы разделения времени используются в режиме диалога с пользователем, поэтому вместо громоздких, детализированных операторов JCL в них используются более простые команды, выполняющие элементарные действия — запуск программы, выдача на экран файла или каталога, копирование или удаление файла и т.п. Пользователю не нужно предвидеть заранее все возможные исходы выполнения команды, гораздо проще увидеть результат выполнения на экране и после этого принять решение, какую команду выполнять следующей. В то же время, некоторые часто повторяющиеся последовательности команд удобно описать один раз в виде «пакетного задания» и затем использовать при необходимости. В этом плане системы разделения времени сохраняют те удобные возможности, которые предоставляли пакетные системы.

Первоначально в качестве аппаратной основы систем разделения времени должны были использоваться «большие» ЭВМ, которые позднее стало принято называть «мейнфреймами» (mainframes). Позднее, по мере прогресса вычислительной техники, это стало по плечу даже миниЭВМ (так назывался в те годы класс компьютеров, занимавших всего лишь один-два небольших шкафчика). Следует особо упомянуть серию миниЭВМ PDP-11, имевшую широчайшее распространение во всем мире в течение полутора десятков лет.

Этот период (70-е годы в мире, 80-е в СССР) характерен глубоким развитием теории и практики создания мощных ОС, содержащих развитые средства управления процессами и памятью, реализующих многопользовательский режим работы. Из большого числа подобных систем особого упоминания заслуживает UNIX — единственная система, благополучно дожившая до нашего времени.

Однозадачные ОС для ПЭВМ

В середине 70-х годов был изобретен микропроцессор, а к началу 80-х микропроцессоры стали догонять по функциональным характеристикам ранее использовавшиеся «большие» процессоры. Эта ситуация сделала почти бесполезным режим разделения времени: зачем делить один процессор между многими задачами и многими пользователями, если проще и дешевле дать отдельный микропроцессор каждому пользователю? Разделение времени осталось целесообразным разве что в отношении суперкомпьютеров.

Появление и бурное распространение персональных компьютеров (ПК) вызвало к жизни новое поколение ОС, которые оказались во много раз проще своих предшественниц. Ненужной оказалась многопользовательская защита. На первых порах показалась ненужной и многозадачность. Все это можно было расценить как явный регресс в развитии ОС.

Наиболее популярной ОС для ранних восьмиразрядных ПК была система CP/M известной тогда фирмы Digital Research, однако с появлением в начале 80-х знаменитой машины IBM PC лидерство было прочно перехвачено системой MS-DOS фирмы Microsoft.

Многозадачные ОС для ПК с графическим интерфейсом

Быстрое развитие технологии привело к тому, что к концу 80-х годов ПК оказались в состоянии решать значительно более сложные и трудоемкие задачи, чем раньше. При этом многие из достижений прежних этапов развития ОС оказались вновь востребованными, но теперь уже в новых условиях, среди которых надо назвать резкое повышение мощности

процессоров и объема памяти, появление высококачественных графических мониторов и развитие сетевых технологий.

Стала реальной такая вещь, как многозадачная ОС для ПК. На смену ОС, которые выполняли текстовые команды, вводимые пользователем с клавиатуры, пришли системы, в которых взаимодействие с пользователем основано на использовании GUI (Graphical User Interface, графический интерфейс пользователя).

Значительная часть ПК работает в составе локальных вычислительных сетей. Это привело к тому, что вопросы защиты данных пользователя вновь приобрели первостепенное значение.

Критерии оценки ОС

При сравнительном рассмотрении различных ОС в целом или их отдельных подсистем возникает вечный вопрос — какая из них лучше и почему, какая архитектура системы предпочтительнее, какой из алгоритмов эффективнее, какая структура данных удобнее и т.п.

Очень редко можно дать однозначный ответ на подобные вопросы, если речь идет о практически используемых системах. Система или ее часть, которая хуже других систем во всех отношениях, просто не имела бы права на существование. На самом деле имеет место типичная многокритериальная задача: имеется несколько важных критериев качества, и система, опережающая прочие по одному критерию, обычно уступает по другому критерию. Сравнительная важность критериев зависит от назначения системы и условий ее работы.

1. Надежность

Этот критерий вообще принято считать самым важным при оценке программного обеспечения, и в отношении ОС его действительно принимают во внимание в первую очередь.

Под надежностью понимается, прежде всего, ее ***живучесть*** операционной системы, т.е. способность сохранять хотя бы минимальную работоспособность в условиях аппаратных сбоев и программных ошибок. Высокая живучесть особенно важна для ОС компьютеров, встроенных в аппаратуру, когда вмешательство человека затруднено, а отказ компьютерной системы может иметь тяжелые последствия.

Во-вторых, способность, как минимум, диагностировать, а как максимум, компенсировать хотя бы некоторые типы аппаратных сбоев. Для этого обычно вводится избыточность хранения наиболее важных данных системы.

В-третьих, ОС не должна содержать собственных (внутренних) ошибок. Это требование редко бывает выполнимо в полном объеме (программисты давно сумели доказать своим заказчикам, что в любой большой программе всегда есть ошибки, и это в порядке вещей), однако следует хотя бы добиться, чтобы основные, часто используемые или наиболее ответственные части ОС были свободны от ошибок.

Наконец, к надежности системы следует отнести ее способность противодействовать явно неразумным действиям пользователя. Обычный пользователь должен иметь доступ только к тем возможностям системы, которые необходимы для его работы. Если же пользователь, даже действуя в рамках своих полномочий, пытается сделать что-то очень странное (например, отформатировать системный диск), то самое малое, что должна сделать ОС, это переспросить пользователя, уверен ли он в правильности своих действий.

2. Эффективность

Эффективность любой программы определяется двумя группами показателей, которые можно обобщенно назвать «время» и «память». При разработке системы приходится принимать много непростых решений, связанных с оптимальным балансом этих показателей.

Важнейшим показателем временно́й эффективности является ***производительность*** системы, т.е. усредненное количество полезной вычислительной работы, выполняемой в единицу времени. С другой стороны, для диалоговых ОС не менее важно ***время реакции*** системы на действия пользователя. Эти показатели могут в некоторой степени противоречить друг другу. Например, в системах разделения времени увеличение кванта времени повышает производительность (за счет сокращения числа переключений процессов), но ухудшает время реакции.

В программировании известна аксиома: выигрыш во времени достигается за счет проигрыша в памяти, и наоборот. Это в полной мере относится к ОС, разработчикам которых постоянно приходится искать баланс между затратами времени и памяти.

Забота об эффективности долгое время стояла на первом месте при разработке программного обеспечения, и особенно ОС. К сожалению, оборотной стороной стремительного увеличения мощности компьютеров стало ослабление интереса к эффективности программ. В настоящее время эффективность является первостепенным требованием разве что в отношении систем реального времени.

3. Удобство

Этот критерий наиболее субъективен. Можно предложить, например, такой подход: система или ее часть удобна, если она позволяет легко и просто решать те задачи, которые встречаются наиболее часто, но в то же время содержит средства для решения широкого круга менее стандартных задач (пусть даже эти средства не столь просты). Пример: такое частое действие, как копирование файла, должно выполняться при помощи одной простой команды или легкого движения мыши; в то же время для изменения разделов диска не грех почитать руководство, поскольку это может понадобиться даже не каждый год.

Разработчики каждой ОС имеют собственные представления об удобстве, и каждая ОС имеет своих приверженцев, считающих именно ее идеалом удобства.

4. Масштабируемость

Термин «масштабируемость» (scalability) означает возможность настройки системы для использования в разных вариантах, в зависимости от мощности вычислительной системы, от набора конкретных периферийных устройств, от роли, которую играет конкретный компьютер (сервер, рабочая станция или изолированный компьютер) от назначения компьютера (домашний, офисный, исследовательский и т.п.).

Гарантией масштабируемости служит продуманная модульная структура системы, позволяющая в ходе установки системы собирать и настраивать нужную конфигурацию. Возможен и другой подход, когда под общим названием объединяются, по сути, разные системы, обеспечивающие в разумных пределах программную совместимость. Примером могут служить версии Windows NT/2000/XP, Windows 95/98 и Windows CE.

В некоторых случаях фирмы, производящие программное обеспечение, искусственно отключают в более дешевых версиях системы те возможности, которые на самом деле реализованы, но становятся доступны, только если пользователь покупает лицензию на более дорогую версию. Но это уже вопрос, связанный не с технической стороной дела, а с маркетинговой политикой.

5. Способность к развитию

Чтобы сложная программа имела шансы просуществовать долго, в нее изначально должны быть заложены возможности для будущего развития.

Одним из главных условий способности системы к развитию является хорошо продуманная модульная структура, в которой четко определены функции каждого модуля и его взаимосвязи с другими модулями. При этом создается возможность совершенствования отдельных модулей с минимальным риском вызвать нежелательные последствия для других частей системы.

Важным требованием к развитию ОС является ***совместимость версий снизу вверх***, означающая возможность безболезненного перехода от старой версии к новой, без потери ранее наработанных прикладных программ и без необходимости резкой смены всех навыков пользователя. Обратная совместимость — сверху вниз — как правило, не гарантируется, поскольку в ходе развития система приобретает новые возможности, не реализованные в старых версиях. Программа из Windows 3.1 будет нормально работать и в Windows XP; наоборот — вряд ли.

Совместимость версий — благо для пользователя, однако на практике она часто приводит к консервации давно отживших свой век особенностей или же просто неудачных решений, принятых в ранней версии системы. В документации подобные архаизмы помечаются

как «устаревшие» (obsolete), но полного отказа от них, как правило, не происходит (а вдруг где-то еще работает прикладная программа, написанная двадцать лет назад с использованием именно этих средств?).

Как правило, наиболее консервативной стороной любой ОС являются не алгоритмы, а структуры системных данных, поэтому дальновидные разработчики заранее строят структуры «на вырост»: закладывают в них резервные поля, используют переменные вместо некоторых констант, устанавливают количественные ограничения с большим запасом и т.п.

6. Мобильность

Под ***мобильностью*** (portability) понимается возможность переноса программы (в данном случае ОС) на другую аппаратную платформу, т.е. на другой тип процессора и другую архитектуру компьютера. Здесь имеется в виду перенос с умеренными трудозатратами, не требующий полной переработки системы.

Свойство мобильности не столь однозначно положительно, как может показаться. Чтобы программа была мобильна, при ее разработке следует отказаться от глубокого использования особенностей конкретной архитектуры (таких, как количество и функциональные возможности регистров процессора, нестандартные команды и т.п.). Мобильная программа должна быть написана на языке достаточно высокого уровня (часто используется язык С), который можно реализовать на компьютерах любой архитектуры. Платой за мобильность всегда является некоторая потеря эффективности, поэтому немобильные системы распространены достаточно широко.

С другой стороны, история системного программирования усеяна останками замечательных, эффективных и удобных, но немобильных ОС, которые вымерли вместе с процессорами, для которых они предназначались. В то же время мобильная система UNIX продолжает процветать четвертый десяток лет, намного пережив те компьютеры, для которых она первоначально создавалась. Примерно 5-10% исходных текстов UNIX написаны на языка ассемблера и должны переписываться заново при переносе на новую архитектуру. Остальная часть системы написана на С и практически не требует изменений при переносе.

Некоторым компромиссом являются ***многоплатформенные*** ОС (например, Windows NT), изначально спроектированные для использования на нескольких аппаратных платформах, но не гарантирующие возможность переноса на новые, не предусмотренные заранее архитектуры.

2. КОНТРОЛЬНЫЕ ВОПРОСЫ

2.1. Дайте определение понятию "операционная система".

2.2. Назовите примеры прикладного, инструментального и системного программного обеспечения.

2.3. Дайте определение понятий "системный вызов", "API", "драйвер", "ядро".

2.4. Какие виды ядер вы знаете? К каким видам относятся ядра известных вам операционных систем?

2.5. Чем ядро отличается от операционной системы?

2.6. Приведите несколько способов классификации операционных систем.

2.7. Назовите требования к современным операционным системам и объясните, что они означают.

2.8. Назовите основные функции операционных систем.

Практическая рабоат №4
РАБОТА С ОСНОВНЫМИ КОМАНДАМИ
В ОПЕРАЦИОННОЙ СИСТЕМЕ (на примере MS-DOS)

Цель работы: научиться пользоваться основными командами операционной системы MS-DOS.

Задание: выполнить работу согласно пункту «Порядок выполнения работы».

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

MS-DOS (англ. *Microsoft Disk Operating System* — дисковая ОС от Microsoft) — коммерческая операционная система фирмы Microsoft для персональных компьютеров. MS-DOS — самая известная ОС из семейства DOS, ранее устанавливаемая на большинство IBM PC-совместимых компьютеров. Со временем она была вытеснена ОС семейства Windows 9x и Windows NT.

MS-DOS была создана в 1981 году и, в ходе её развития, было выпущено восемь крупных версий (1.0, 2.0 и т. д.) и два десятка промежуточных (3.1, 3.2 и т. п.), пока в 2000 году Microsoft не прекратила её разработку. Это был ключевой продукт фирмы, дававший ей существенный доход и маркетинговый ресурс, в ходе развития Microsoft от разработчика языка программирования до крупной компании, производящей самое разнообразное программное обеспечение.

Последняя официальная версия 6.22. Однако существует версия 7.1 в виде ядра Windows 98, которая загружается на начальном этапе загрузки системы.

Когда компьютер включается, то на экране появляются быстро сменяющие друг друга сообщения. После того, как вывод сообщений закончится, на экране появится системное приглашение вида*:*

C:\>,

где С – имя диска, с которого происходит загрузка операционной системы.

Мигающий символ подчеркивания, находящийся после системного приглашения, называется курсором. Он показывает место ввода команды. Строка, в которой набирается команда, называется командной строкой.

После набора команды необходимо нажать клавишу **Enter**.

Набирать команду можно как строчными, так и прописными буквами.

Просмотр содержимого каталога. Для просмотра содержимого каталога необходимо в командной строке набрать **dir**. На экране будет выдан текст, который называется списком каталога. Список каталога - это список всех файлов и подкаталогов, содержащихся в данном каталоге. Все имена, имеющие пометку

, относятся к каталогам. Чтобы просмотреть список файлов другого каталога, следует сначала сменить каталог, а затем снова воспользоваться командой DIR.

Если в каталоге большой список файлов, то этот список промелькнет слишком быстро, чтобы его можно было рассмотреть. Для того чтобы информация выдавалась последовательно, по одному файлу на экране существует команда **dir /p**.

После вывода первого экрана будет сделана пауза. Для продолжения просмотра необходимо нажать любую клавишу. Эту операцию необходимо повторять до тех пор, пока не появится системное приглашение.

С помощью команды **dir** можно просмотреть список файлов каталога в широком формате. Это осуществляется с помощью команды **dir /w**.

Появится многоколоночный список файлов каталога. В этом случае перечисляются только имена файлов. Информация о размере файла, дате и времени создания не выдается.

Если каталог содержит файлов больше, чем может поместиться на экране, то можно использовать команду просмотра содержимого каталога, указав оба параметра **DIR /W/P**.

Переход на другой диск. Для перехода к другому диску и просмотра находящихся на нем файлов необходимо вставить дискету в дисковод A и набрать в командной строке **A:** и нажать клавишу Enter. Системное приглашение теперь будет выглядеть так: **A:**\>

Для просмотра списка файлов на дискете, которая установлена в дисковод, необходимо набрать команду DIR. Появится список файлов, находящихся на дискете.

Чтобы вернуться к диску С необходимо набрать **C:**.

Переход к указанному диску осуществляется, когда в командной строке набирается буква диска, после которой ставится двоеточие и нажимается клавиша Enter. Буква диска в системном приглашении показывает, какой диск является текущим диском. Все вводимые команды выполняются над текущим диском и текущим каталогом, если другой диск или каталог не указаны специально.

Для просмотра файлов на диске, не являющимся текущим, используется также команда DIR, после которой через пробел указывается имя диска. Например, просмотр содержимого диска A: при текущем диске C: осуществляется вводом следующей команды **DIR A:.**

Имя диска A:, введенное после команды DIR, является параметром, указывающим над каким диском выполняется команда.

Смена каталога. Смена каталога осуществляется с помощью команды **CD**. Допустим нам необходимо просмотреть содержимое каталога DOS, находящегося на диске C:. Для этого необходимо в командной строке набрать следующую команду: **cd dos**.

Системное приглашение изменится и будет выглядеть следующим образом:

C:\DOS>

Системное приглашение показывает, какой каталог является текущим. В данном случае смена каталога прошла успешно, так как теперь системное приглашение содержит имя каталога DOS, который стал текущим. Для просмотра списка файлов в каталоге DOS используется также команда DIR.

Для перехода в корневой каталог используется команда **CD ** и системное приглашение теперь будет выглядеть **C:\>.**

Вне зависимости от того, какой каталог является текущим в данный момент, эта команда всегда возвратит к корневому каталогу диска. Корневой каталог не имеет имени, ссылка на него происходит с помощью символа обратный слэш **(\)**.

Создание каталога. Создание каталога полезно, если необходимо объединить связанные по смыслу файлы в группы. Для создания каталога используется команда **MD**. Например, для создания каталога с именем TEXT в командной строке следует набрать **MD TEXT.**

Для того чтобы перейти в этот каталог, то есть сделать его текущим, необходимо набрать **CD TEXT.**

Системное приглашение изменится и будет выглядеть следующим образом: **C:\TEXT>**

Для того чтобы в каталоге TEXT создать подкаталог PISMO, необходимо в командной строке набрать **MD PISMO**.

Чтобы убедиться, что каталог создан, необходимо набрать команду DIR. Чтобы каталог PISMO стал текущим, в командной строке необходимо набрать команду **CD PISMO**.

Системное приглашение теперь должно выглядеть следующим образом: **C:\TEXT\ PISMO>**

Чтобы вернуться назад к каталогу TEXT, то есть сделать его текущим, вводится команда **CD..**

Системное приглашение теперь должно выглядеть следующим образом **C:\TEXT>**

Когда команда CD сопровождается двумя точками (**..**), система перемещается на один уровень вверх в структуре каталогов. В данном случае мы переместились на один уровень вверх - из каталога PISMO в каталог TEXT.

Копирование файлов. При копировании файлов используется команда **COPY** с двумя параметрами: источником (откуда копируется файл и имя этого файла) и назначением (куда следует поместить копию). Команда COPY имеет следующий формат

COPY источник назначение

Например, для копирования файла EDIT. HLP из каталога DOS, находящемся на диске C: в каталог TEXT, который находится также на диске C: команда будет иметь следующий вид:

COPY C:\DOS\EDIT. HLP C:\TEXT

Появится сообщение “1 файл скопирован”, т. е. копия файла EDIT. HLP будет находиться в каталоге TEXT диска C:.

В процессе работы может возникнуть необходимость в копировании не одного файла, а сразу нескольких файлов.

Для просмотра группы файлов на диске также используется команда DIR, в качестве параметра которой указывается шаблон имени файла **(*** или **?).** Например, если необходимо вывести на экран список всех файлов, имеющих расширение. txt, то команда DIR будет выглядеть следующим образом DIR *.txt. Это означает, что на экран будут выведены все файлы

с расширением. txt. Звездочка замещает собой все символы в имени файла до точки, отделяющей имя файла от его расширения.

Для того чтобы скопировать все файлы с данным расширением в каталог TEXT, в командной строке следует набрать следующее: **COPY *.txt C:\TEXT.**

Эта команда скопирует все файлы из текущего каталога в каталог TEXT.

Переименование файлов. Для того чтобы переименовать существующий файл используется команда **REN**. Эта команда требует задания двух параметров. Первый параметр - это имя существующего файла, которое нужно изменить, а второй - новое имя этого файла. Параметры разделяются пробелом. Команда будет иметь следующий вид:

REN старое_имя новое_имя. Например, REN NAME1.TXT NAME2.TXT.

Удаление файлов. Для удаления файлов используется команда **DEL** (DELETE). В командной строке набирается команда DEL и через пробел имя удаляемого файла. Для того, чтобы убедиться, что файл действительно удален, используем команду DIR. В появившемся списке файлов удаленного файла быть не должно.

Для удаления группы файлов используется также команда DEL, но после пробела указывается шаблон имен файлов. Допустим, нам необходимо удалить все файлы с расширением. txt. Команда будет выглядеть следующим образом: **DEL *.txt.**

Чтобы убедиться, что все файлы с расширением. txt удалены, используется команда DIR.

Удаление каталога. В процессе работы может возникнуть ситуация, когда каталог необходимо удалить. Для удаления каталога используется команда **RD**. Для удаления подкаталога PISMO в командной строке набирается команда -> **RD PISMO**.

Набрав команду DIR, можно убедиться, что каталог удален, так как в списке на экране он отсутствует.

Удалить каталог, являющийся текущим и содержащим внутри себя файлы или другие подкаталоги невозможно. Перед удалением такого каталога необходимо удалить всю имеющуюся в нем информацию и перейти на уровень вверх, то есть в командной строке нужно набрать команду CD..

Использование справки. Справочная система MS DOS содержит полную информацию о командах MS DOS. Запустить систему справки можно двумя способами - с выводом на экран общего оглавления для выбора интересующего раздела либо сразу с выводом нужного раздела.

Для запуска справочной систему и выбора нужного раздела из общего оглавления необходимо в командной строке набрать **HELP**. На экране появится общее оглавление. Для выбора нужного раздела необходимо нажать на клавиатуре клавишу, соответствующую первой букве названия нужной команды. Будет выделена первая команда, которая начинается на эту букву. Если это не та команда, которая нужна, то необходимо продолжить нажимать клавишу до тех пор, пока нужная команда не будет выделена, а затем нажать клавишу Enter. По списку разделов оглавления можно также перемещаться с помощью клавиш перемещения курсора, клавиши **Tab**, клавиш **PageUp, PageDown**.

Для запуска справочной системы и получения информации о конкретной команде в командной строке необходимо набрать HELP, а затем через пробел название необходимой команды. Например, чтобы получить справку по команду COPY, в командной строке необходимо набрать HELP COPY.

Краткую информацию о назначении и использовании команды можно получить без запуска справочной системы MS DOS. Эта информация менее подробна, чем выдаваемая справочной системой MS DOS. Для получения краткой справки в командной строке набирается имя команды, а затем - пробел и переключатель **/?**. Например, чтобы просмотреть синтаксис команды DIR, в командной строке надо набрать -> **DIR /?**.

Восстановление удаленных файлов. Для восстановления ошибочно удаленных файлов имеется программа **UNDELETE**, которая работает следующим образом:

- каталог, который содержит ошибочно удаленные файлы необходимо сделать текущим. Например, каталог DOS. Для этого в командной строке необходимо набрать CD \DOS;

- далее необходимо набрать UNDELETE;

- по мере обнаружения удаленных файлов программа перечисляет их по одному и выдает запрос, нужно ли восстанавливать указанный файл. Чтобы восстановить файл, нажимается клавиша Y. Программа может также попросить задать первую букву имени файла.

Для получения справки о работе данной программы в командной строке надо набрать **HELP UNDELETE**.

Настройка системы. Значительная часть информации о настройке системы хранится в двух файлах, находящихся в корневом каталоге диска, с которого производится загрузка:

- файл **CONFIG. SYS** содержит команды, которые касаются конфигурации аппаратуры компьютера (память, клавиатура, мышь, принтер и т. д.). Когда система MS DOS загружается, то сначала выполняются инструкции, находящиеся в файле CONFIG. SYS.

- файл **AUTOEXEC. BAT** - это файл пакетной обработки, который система выполняет сразу после обработки файла CONFIG. SYS. Файл AUTOEXEC. BAT содержит команды, которые требуется выполнять при запуске системы.

Операционная система MS DOS выполняет команды, содержащиеся в файлах CONFIG. SYS и AUTOEXEC. BAT при каждой загрузке компьютера. При желании, есть возможность указать системе, что команды в этих файлах следует пропустить.

При установке системы операционной системы MS DOS программа установки (SETUP) формирует базовую конфигурацию системы, которая работоспособна для большинства компьютеров. Однако эту конфигурацию можно изменить с помощью команд, находящихся в файлах CONFIG. SYS и AUTOEXEC. BAT.

Для редактирования файлов CONFIG. SYS и AUTOEXEC. BAT следует использовать текстовый редактор, который может сохранять файлы в виде текста. Операционная система MS DOS считывает содержимое файлов CONFIG. SYS и AUTOEXEC. BAT только при загрузке компьютера. Это означает, что для того, чтобы изменения, сделанные в указанных файлах, имели действие, следует перезагрузить компьютер (то есть нажать комбинацию клавиш **CTRL-ALT-DEL** горячая (локальная) перезагрузка).

Перед тем, как вносить изменения в файлы CONFIG. SYS и AUTOEXEC. BAT, необходимо сделать резервные копии изменяемых файлов (скопировать их на дискету), так как установки, находящиеся в этих файлах, контролируют основные компоненты системы. Если изменения сделаны некорректно, то система может не загрузиться. В этом случае необходимо поместить дискету в дисковод A: и перезагрузить компьютер (нажать клавиши CTRL-ALT-DEL).

Для того, чтобы сформировать дискету с резервными копиями изменяемых файлов, необходимо вставить дискету в дисковод A и набрать в командной строке

FORMAT A: /S

После выполнения данной команды происходит форматирование дискеты с переносом на нее системных файлов. Далее копируются файлы CONFIG. SYS и AUTOEXEC. BAT на этот диск, который станет загрузочным, с помощью следующих команд:

COPY C:\CONFIG. SYS A:

COPY C:\AUTOEXEC. BAT A:

Norton Commander (NC) — популярный файловый менеджер для DOS, первоначально разработанный американским программистом John Socha. (Некоторые дополнительные компоненты были полностью или частично написаны другими людьми: *Linda Dudinyak* — Commander Mail, вьюверы; *Peter Bradeen* — Commander Mail; *Keith Ermel, Brian Yoder* — вьюверы.) Программа была выпущена компанией Peter Norton Computing (глава — Питер Нортон), которая позже была приобретена корпорацией Symantec.

Начиная с версии 4.0 программу разрабатывала целая команда программистов, поскольку в 1990 фирма Peter Norton Computing, была куплена компанией Symantec, но новый коммандер постепенно начал утрачивать популярность, так как увеличил размер занимаемой памяти (что было критично для DOS), содержал ошибки и, к тому же, начал вытесняться собственными клонами. Менее функциональные, чем NC, Volkov Commander и Pie Commander, более-менее точно копировали нортоновский интерфейс. DOS Navigator, визуально схожий с Norton

Commander-ом, предоставлял гораздо больший ряд возможностей. Впоследствии клоны появились и на других операционных системах: BSD, GNU/Linux — Midnight Commander, Krusader; Microsoft Windows — FAR Manager, Total Commander; и другие аналогичные программы.

2. ПОРЯДОК ВЫПОЛНЕНИЯ ЗАНЯТИЯ

1. Отформатировать дискету, для чего открыть на рабочем столе «Компьютер», выбрать Диск 3.5’’A, щелкнуть правой кнопкой мыши и выбрать команду «Форматировать».
2. После удачного завершения форматирования выполнить команду: Пуск -> Все программы -> Стандартные - > Командная строка.
3. Перейти в корневой каталог устройства C:, для чего выполнить команду: CD\.
4. Создать на диске **C:** каталог с именем **FIO** по первым буквам ваших фамилии, имени и отчества в латинском регистре. Для этого нужно задать команду **MD FIO**.
5. Создать в каталоге **FIO** на диске **C:** текстовый файл **PRIMER.TXT**, введя в него текст с клавиатуры:

   задать команду **COPY CON: C:\FIO\PRIMER.TXT**
6. Ввести следующий текст:

   «Одной из распространенных операционных систем для персональных компьютеров является MS-DOS, дисковая операционная система, разработанная корпорацией Microsoft.

   Три основные функции операционной системы MS-DOS: Обмен данными между компьютером и различными периферийными устройствами (терминалами, принтерами, гибкими дисками, жесткими дисками и т.д.). Такой обмен данными называется вводом-выводом данных.

   Обеспечение системы организации и хранения файлов. Загрузка программ в память и обеспечение их выполнения.»

   Закончить ввод текста, нажав клавиши **Ctrl+Z**, затем клавишу **Enter**. После этого создание текстового файла будет завершено.
7. Просмотреть созданный файл на экране видеомонитора: задать команду **TYPE C:\FIO\PRIMER.TXT.**
8. Установить текущим диск **A:**

   задать команду: **A:**
9. Просмотреть корневой каталог диска **A:**

   задать команду: **DIR A:**
10. Создать на диске **A:** каталог **111**:

    задать команду: **MD 111.**

    2.11. Просмотреть корневой каталог диска **A:**

    задать команду: **DIR**

    2.12. Просмотреть содержимое каталога **111** диска **A:**

    задать команду: **DIR 111**

    2.13. Скопировать файл **PRIMER.TXT** из каталога **C:\FIO** в **A:\111**:

    задать команду **COPY C:\FIO\PRIMER.TXT A:\111\PRIMER.TXT.**
14. Просмотреть содержимое каталога **111** диска **A:**

    задать команду: **DIR 111**
15. Создать на диске **A:** каталог **222**:

    задать команду: **MD 222.**
16. Скопировать файл **PRIMER.TXT** из каталога **C:\FIO** в **A:\222**:

    задать команду: **COPY C:\FIO\PRIMER.TXT A:\222\PR1MER.TXT**
17. Просмотреть структуру каталогов на дисках **A:** и **C:**

    задать команды: **TREE A:, TREE C:.**
18. Установить текущим каталог **A:\222:**

    задать команду: **CD 222**
19. Переименовать файл **PRIMER.TXT** в каталоге **A:\222** в **PROBA.DOC:**

    задать команду: **REN PRIMER.TXT PROBA.DOC**

20. Просмотреть содержимое каталога **222** диска **A:**
задать команду: **DIR**
21. Установить текущим корневой каталог диска **A:**
задать команду: **CD **
22. Скопировать файл **PROBA.DOC** из каталога **222** на диске **A:** в каталог **111**
задать команду: **MOVE A:\222\ PROBA.DOC A:\111\ PROBA.DOC**
23. Просмотреть содержимое каталогов **222** и **111** на диске **A:**
задать команды: **DIR A:\222**
DIR A:\111
24. Удалить файл **PROBA.DOC** из каталога **A:\222**:
задать команду **DEL A:\222\PROBA.DOC**
25. Попытаться удалить каталог **111**с диска **A:**
задать команду: **RD** 111
Примечание. Удалить непустой каталог не удастся.
26. Удалить все файлы из каталога A**:\111**:
задать команду: **DEL A**:\111*.*
Подтвердить удаление всех файлов, нажав клавишу **Y**.
27. Удалить каталог **111**с диска **A**:
задать команду: **RD** 111
28. Просмотреть корневой каталог диска **A:**
задать команду: **DIR**
29. Очистить экран видеомонитора:
задать команду: **CLS**
30. Показать текущую дату в компьютере и заменить ее на предыдущую:
задать команду: **DATE**
Изменить день.
31. Показать текущее время в компьютере и заменить его на час позже:
задать команду: **TIME**
Изменить час.
32. Создать на диске **A:** каталог **333** и скопировать в него файлы, имеющие первый символ **«P»** в расширении из каталога **C:\TP**;
задать команду: **MD A:\333**
задать команду **COPY** C**:\TP*.P* A:\333*.***
33. Записать в файл **TR_FILE.TXT** структуру каталогов на диске **A:** со списком файлов по каталогам:
задать команду: **TREE A:/F > TR_FILE.TXT**
34. Восстановить прежние дату и время:
задать команду: **DATE** , изменить день задать команду: **TIME** , изменить час.
2.35. Удалить с диска **A:** каталог **333** и все файлы в нем:
задать команду: **DELTREE 333**
подтвердить удаление, нажав клавишу **«Y».**

2. КОНТРОЛЬНЫЙ ТЕСТ

1. Какая из предложенных записей означает полное имя файла?
    1. A :/KATALOG/KATALOG1 /text.txt
    2. A:\KATALOG\KATALOGl\text.txt*
    3. A:\KATALOG\KATALOG1 \> text.txt
2. Какое имя файла является правильным?
    1. TXT.PRN*
    2. PRN.TXT
    3. CON.PRN
3. Какое имя файла является недопустимым в ОС MS-DOS?

1. TEXT l.TXT*
2. PPJCOL.TXT
3. RISUNOK.TXT

4. Какое кол-во символов может быть в имени файла в ОС MS-DOS:
   1. max 3
   2. max 8*
   3. max 255
   4. 0...8
   5. 0 ... 3
5. Имя файла text.l допустимо или нет?
   1. Нет.
   2. Да.*
6. Что означает системное приглашение C:\DOS\Dl> ?
   1. Пользователь находится в подкаталоге D1 каталога DOS устройства C:*
   2. Пользователь находится в корневом каталоге устройства C:
   3. Пользователь находится в каталоге DOS подкаталога D1 устройства C:
7. Укажите имя файла, которое неверно?
   1. Gr1.prn
   2. Super.com
   3. SuperCalc.exe*
8. Какое имя файла является правильным?
   1. $katalog.txt*
   2. speeddisk.exe
   3. {katalog}.txt
9. Что означает символ "*" при задании маски для обозначения группы файлов?
   1. один символ
   2. 256 символов
   3. любое количество символов (max 8)*
10. Что означает символ "?" при задании маски для обозначения группы файлов?
    1. любое количество символов
    2. один символ*
    3. 8 символов
11. В каталоге имеются следующие файлы: ncd.exe, ndd.exe, norton.exe, ndiags.exe, ncc exe.

Как следует задать маску для обозначения группы файлов:
ncd.exe, ndd.exe, ncc.exe?
    1. n??.exe*
    2. nc?.exe
    3. n*.exe
    4. n*.*
    5. n?*.*
12. Какое количество символов может быть использовано для написания расширения в обозначении файла?
    1. 1 ... 8
    2. 1 ... 3
    3. 0 ... 3*
13. В каталоге имеются следующие файлы: filefind.exe, fl.exe, fs.exe, fa.exe, ffa.exe.

Как следует задать маску для обозначения группы файлов:
fa.exe, ffa.exe, fl.exe, fs.exe?

1. F*.*
2. f*.exe
3. f?.exe
4. f??.*
5. f*?.exe*

14. В каталоге с имеются следующие файлы:
Speeddisk.exe, Sformat.exe, Smartcan .exe, Sysinfo.exe, St.exe.
Пользователь задал маску для обозначения группы файлов S??????.*
Какие имена файлов при этом окажутся выделены цветом?
    1. все файлы
    2. sysinfo.exe, sformat.exe и st.exe*
    3. sysinfo.exe и sformatexe
    4. st.exe
    5. speedisk.exe и smartcan.exe
15. Что обозначает маска E???A. * ?
    1. Имя файла начинается с буквы E , заканчивается буквой A

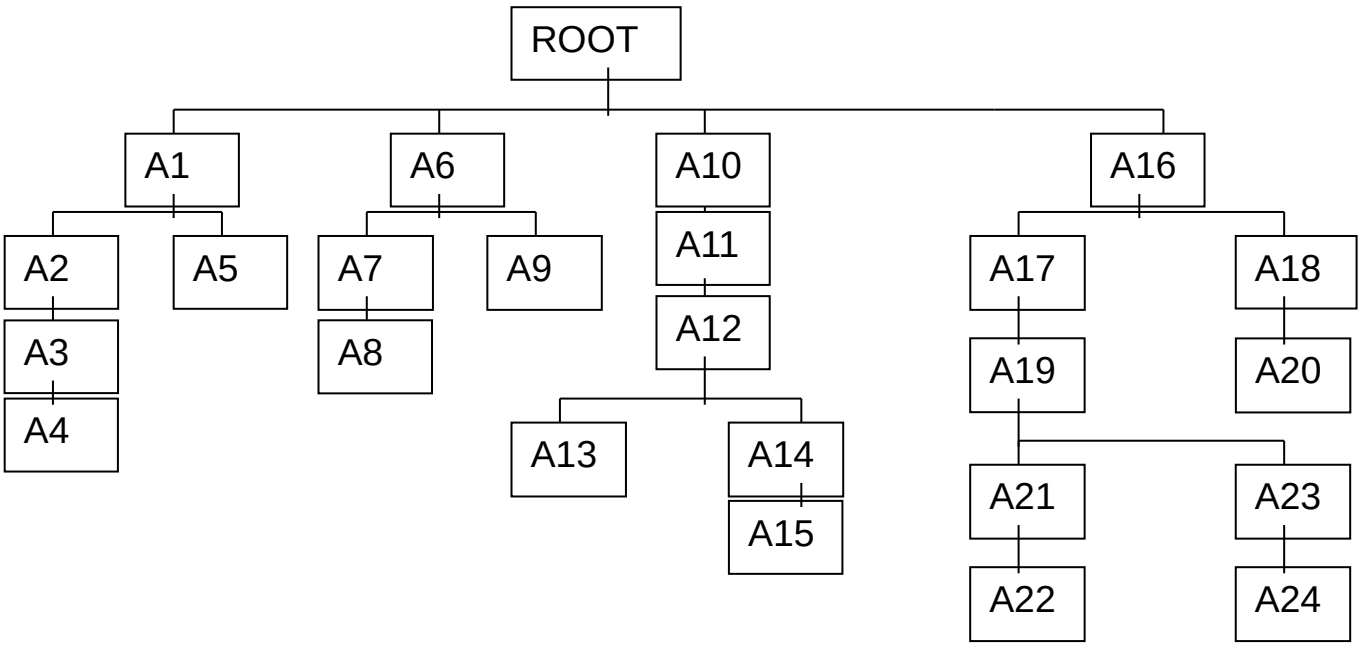
1. Имя файла начинается с буквы E, заканчивается буквой A, состоит из 5 символов и более, и имеет любое расширение
2. Имя файла начинается с буквы E, заканчивается буквой A, состоит из 5 символов и менее и имеет любое расширение*

16.Написать маску для обозначения следующей группы файлов:
Имя файла начинается с буквы D, в имени файла может быть max до 8 символов, расширение начинается с буквы h .
1. D*.h*.
2. *?D.h?
3. D????.h?*
4. D?*.h?*
5. D*.h*(*)

3. Отчет по выполненной работе

Отчет должен содержать перечень основных команд и их назначение.

Практическая работа №5
Тема: Работа с каталогами MS-DOS



Задание:

1. Создать и проверить дерево каталогов
2.Переименовать A6 в «KATALOG»
3. Содержимое A19 копировать в A9
4. A8 переместить в A3
5. Удалить A12
6. Удалить A2

1. Создать и проверить дерево каталогов

```
md root
cd root
md A1
cd A1
md A2
cd A2
md A3
cd A3
md A4
cd\root\A1
md A5
cd\root
md A6
cd A6
md A7
md A9
cd A7
md A8
cd\root\
md A10
cd A10
md A11
cd A11
md A12
cdA12
md A13
md A14
cd A14
md A15
cd\root\
md A16
cd A16
md A18
md A17
cd A17
md A19
cd A19
md A21
```

mdA23
cdA23
mdA24
cd..
cd..
cd A21
md A22
cd..
cd..
cdA18
mdA20
tree

2.Переименовать А6 в «KATALOG»
S:\root>
ren A6 KATALOG
cd..

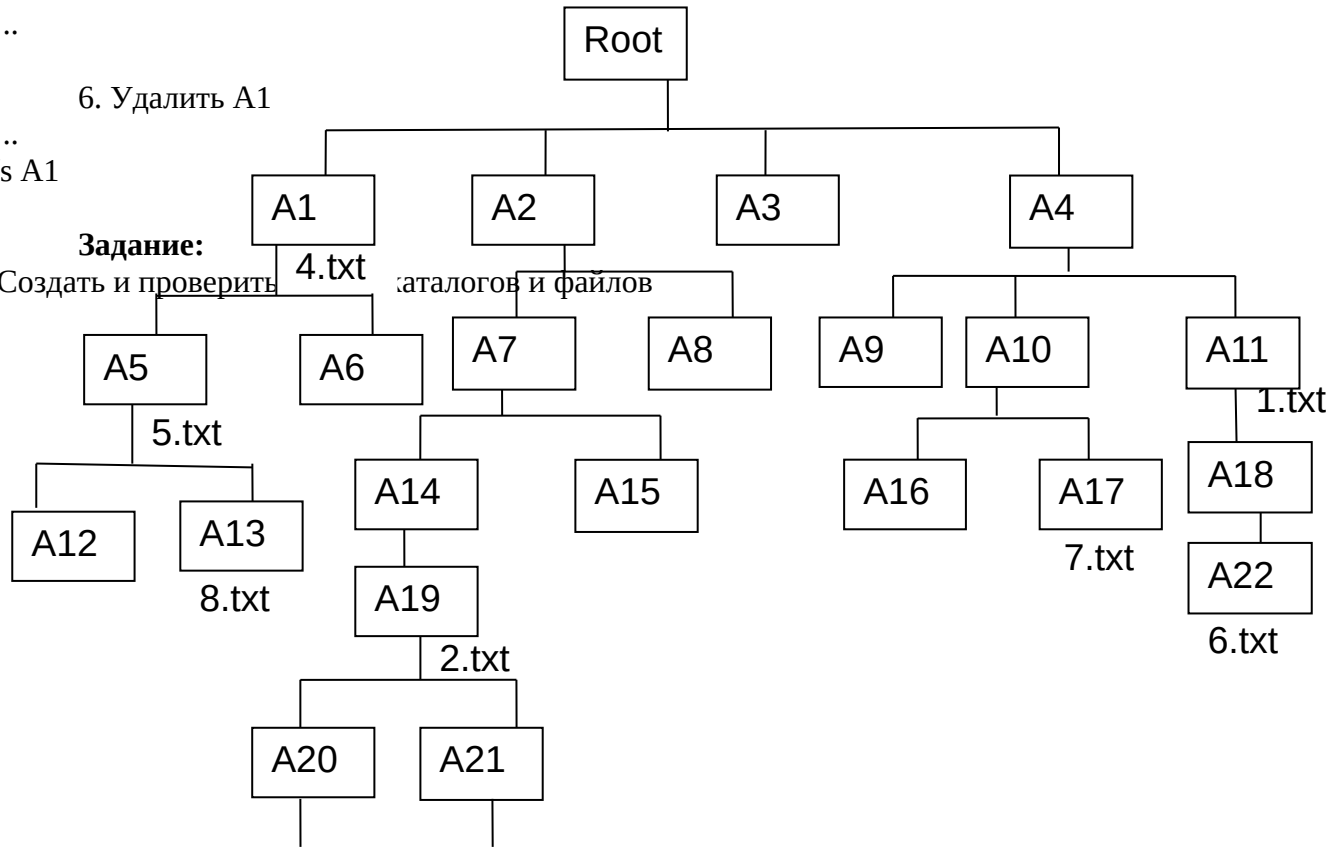
3. Содержимое А19 копировать в А9
cd A16
cd A17
cd A19
xcopy/e A19 S: \root\ A16\A17\A19
cd\root\

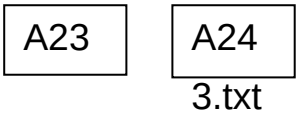
4. А8 переместить в А3
move A8 S:\root\ A3
cd..

5. Удалить А12.
S: \root>
cd A10
cd A11
rd A12
cd ..

6. Удалить А1
cd ..
rd/s A1

Задание:
1. Создать и проверить [illegible]аталогов и файлов





4.txt (цу)
5.txt (В группе По-213)
1.txt (ол)
7.txt (дпл)
8.txt (Я учусь)
2.txt (аро)
6.txt (В УРТТ имени А.С Попова)
3.txt (Сутырин Александр Сергеевич)

2.Обьеденить файлы 3.txt, 8.txt, 6.txt, 5.txt
3.Переименновать 4.txt в 444.txt
4. Скопировать A11 в A23
5. 7.txt переместить в A24
6. Удалить каталог A19

1. Создать и проверить дерево каталогов и файлов
md root
cd root
md A1
cd A1
copy con 4.txt
цу (F6 и Enter)
md A5
md A6
cd A5
copy con 5.txt
В группе По-213 Ctrl+Z
md A7
cd A7
md A8
md A9
cd A9
md A10
cd..
cd..
cd..
cd..
md A2
cd A2

```
md A11
md A12
cd A11
copy con 1.txt
 ол (F6 и Enter)
md A13
cd A13
copy con 8.txt
 Я учусь (F6 и Enter)
cd..
cd A12
md A14
cd A14
md A15
md A16
cd A16
md A17
cd A17
copy con 7.txt
 дтл (F6 и Enter)
cd..
cd..
cd..
cd..
cd..
md A3
cd A3
md A8
cd..
md A4
cd A4
md A19
md A20
cd A19
copy con 2.txt
 apo (F6 и Enter)
md A21
cd A21
md A23
cd A23
md A24
cd A24
copy con 3.txt
Сутырин Александр Сергеевич (F6 и Enter)
cd..
cd..
cd..
cd..
cd..
md A3
cd..
md A4
```

cd A4
md A9
md A10
md A11
cd A11
md A18
cd A18
md A 22
cd A22
copy con 6.txt
В УРТТ им., А.С.Попова (F6 и Enter)
cd..
cd..
cd..
cd A10
md A17
md A16
cd..
cd..
tree

2. Объединить файлы 3.txt, 8.txt, 6.txt, 5.txt

cd..
cd..
cd..
cd..
cd..
copy con S:\root\A4\A19\A21\A23\A24+S:\root\A2\A11\A13+S:\root\A1\A5+ S:\root\A4\A19\A22 S:\root\of.txt
type of .txt
Я учусь в УРТТ имени А.С.Попова в группе По-213 Сутырин Александр Сергеевич

3. Переименовать 4.txt в 444.txt

cd\root\a1
ren 4.txt 444.txt

4. Скопировать А11 в А23

cd\root\a9
copy A11 s:\root\a15\a17\a23

5. Переместить 7.txt в А24

cd\root\a4\a10\a17
move 7.txt s:\root\a2\a7\a14\a19\a21\a24

6. Удалить А19

cd\root\a2\a7\a14
rd/s A19

Практическая работа № 6
КОМАНДНЫЕ ФАЙЛЫ ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS

Цель занятия: научиться пользоваться основными командами операционной системы Windows.

Задание: составить конспект основных структур командного файла, составить командный файл для загрузки системы в минимальной конфигурации, следуя инструкциям описания.

1. **ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ**

1. **Общие сведения о командном процессоре Windows**

Командные файлы (скрипты, сценарии, батники) - это обычные текстовые файлы с расширением **.bat** или **.cmd**, строки которых представляют собой специальные команды или имена исполняемых файлов. Строки командных файлов обрабатываются специальной программой - командным процессором операционной системы, часто называемым интерпретатором команд. Для операционных систем DOS и Windows9X в качестве интерпретатора команд используется **command.com**, для Windows NT и старше - **cmd.exe**.

Строки командных файлов могут содержать специфические команды самого процессора команд (FOR, ECHO, REM и т.п.) или имена исполняемых модулей (net.exe, regedit.exe, sc.exe) Командный процессор может быть запущен в интерактивном режиме через **Пуск - Выполнить - CMD.EXE**. В данном режиме, вы увидите окно консоли с приглашением к вводу команд. Возможный список большинства консольных команд можно получить введя: **HELP.** Справочную информацию по конкретной команде можно получить, указав ее название в качестве параметра команды HELP: **HELP Имя команды**.

Если работа осуществляется в русифицированной версии Windows, то учтите, что в среде командного процессора символы национального алфавита используются в DOS-кодировке. Для переключения между кодовыми страницами Windows и DOS используется команда

CHCP номер страницы:

CHCP 866 - использовать кодовую страницу 866 (DOS);

CHCP 1251 - использовать кодовую страницу 1251 (WINDOWS).

Для просмотра и редактирования командных файлов, содержащих символы русского алфавита нужно использовать редактор с поддержкой DOS-кодировки. Если вы используете стандартное приложение "Блокнот" (notepad.exe), то для правильного отображения символов русского алфавита нужно выбрать шрифт **Terminal**, с помощью меню **Правка - Шрифт...** Внешний вид окна CMD.EXE (консоли Windows) можно изменить с помощью команды **COLOR.**

В качестве аргументов для команды используются 2 шестнадцатеричные цифры, задающие цвет фона и цвет символа.

COLOR F0 - черные символы на белом фоне.

COLOR 0E - светло-желтые символы на черном фоне.

HELP COLOR - подсказка для команды COLOR.

Работа с командным процессором предполагает использование двух устройств - устройства ввода (клавиатуры) и устройства вывода (дисплей). Однако, имеется возможность изменить стандартно используемые устройства ввода-вывода с помощью специальных символов **- символов перенаправления:**

> - перенаправление вывода

< - перенаправление ввода

Для вывода справки не на экран а, например, в файл с именем help.txt, можно использовать следующую команду:

HELP > help.txt.

При выполнении данной команды, в текущем каталоге будет создан файл с именем help.txt, содержимым которого будет результат вывода команды HELP. Если файл help.txt существовал на момент выполнения команды, его содержимое будет перезаписано. Для того чтобы дописать данные в конец существующего файла, используют удвоение символа перенаправления вывода - **">>".** Например:

HELP GOTO > myhelp.txt - в файл myhelp.txt будет выдана справка по использованию команду GOTO;

HELP COLOR >> myhelp.txt - в конец файла myhelp.txt будет дописана справка по использованию команды COLOR.

Простейший пример перенаправления ввода:

cmd.exe < commands.txt - командный процессор не будет ожидать ввода команд с клавиатуры, а считает их из файла commands.txt.

При запуске командного процессора можно указать конкретную команду в качестве аргумента командной строки:

cmd.exe /C HELP FOR - выполнить команду HELP FOR и завершиться (ключ /C);

cmd.exe /K HELP FOR - выполнить команду HELP FOR и перейти в режим ожидания дальнейшего ввода команд (ключ /K).

Подробную справку по использованию cmd.exe можно получить, введя в качестве аргумента ключ **/?**

cmd.exe /?

Кроме символов перенаправления ввода-вывода в командной строке могут использоваться символы объединения команд - **&&** и **|| :**

cmd.exe /C "HELP IF > nul" && Echo HELP Executed || Echo HELP Not Executed - выполнить команду HELP IF и при успешном результате выполнить команду Echo HELP Executed, а при неуспешном - Echo HELP Not Executed. Команды, объединяемые для выполнения с помощью конструкции **&&** , не нужно заключать в двойные кавычки. Выполнение строки cmd.exe /C "HELP IF > nul" && Echo HELP Executed || Echo HELP Not Executed завершится сообщением HELP Executed, а выполнение cmd.exe /C "HELP uIF > nul" && Echo HELP Executed || Echo HELP Not Executed где неверно задан аргумент команды HELP (uIF), завершится сообщением HELP Not Executed.

Файлы с расширением .bat или .cmd в среде Windows стандартно открываются командным процессором аналогично примеру, где список команд считывается не с устройства ввода, а из текстового файла.

2. **Использование переменных в командных файлах**

Существует такое понятие, как **переменные окружения (environments)** - это переменные, значения которых характеризуют среду, в которой выполняются команда или пакетный файл. Иногда их называют переменными среды. Принимаемые значения этих переменных формируются при загрузке, регистрации пользователя в системе, старте или завершении некоторых приложений, и, кроме того, могут быть заданы с помощью специальной команды **SET:**

SET переменная=строка

где: **переменная** - имя переменной среды;

строка - строка символов, присваиваемая указанной переменной.

Например, командная строка

SET myname=Vasya

создает переменную **myname**, принимающую значение **Vasya**.

Значение, присвоенное какой-либо переменной, доступно для обработки в командных файлах, при использовании ее имени, заключенного в знаки процента - **%** . Например команда выдачи текста на дисплей ECHO в виде:

ECHO date - выведет на экран слово "date", а команда **ECHO %date%** выведет на экран значение переменной date - текущую дату в формате операционной системы.

С помощью команды SET обычно задается и модифицируется путь поиска исполняемых программ - переменная окружения **PATH**:

SET PATH=C:\Windows;C:\windows\system32

После выполнения данной команды, поиск исполняемых файлов будет выполняться в каталоге **C:\Windows**, и, если результат неуспешен, в **C:\windows\system32.**

Допустим необходимо выполнить программу **myedit.exe**, размещенную в каталоге **C:\ NewProgs.** Если в командной строке не задан полный путь, а только имя исполняемого файла - **myedit.exe,** то сначала будет выполняться поиск файла myedit.exe в текущем каталоге, и если

он не будет найден - в каталогах, список которых задается значением переменной PATH. Символ «**;**» является разделителем элементов в списке путей поиска. Если в приведенном примере, текущим каталогом не является **C:\NewProgs**, и в остальных каталогах, заданных значением переменной PATH. нет исполняемого файла **myedit.exe**, то запуск приложения myedit.exe завершится ошибкой. Однако если есть необходимость его запуска без указания полного пути и при любом значении текущего каталога, можно модифицировать значение переменной PATH.

Команда **SET PATH=C:\NewProgs;%path%** изменит текущее значение PATH, добавив каталог C:\NewProgs в начало списка. Выполнение команды SET без параметров позволяет получить текущие значения переменных окружения:

NUMBER_OF_PROCESSORS=1 - количество процессоров
OS=Windows_NT- тип ОС
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Far - путь поиска исполняемых файлов.
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH - расширения для исполняемых файлов.
PROCESSOR_ARCHITECTURE=x86 - архитектура процессора.
PROCESSOR_IDENTIFIER=x86 Family 6 Model 8 Stepping 1, AuthenticAMD - идентификатор процессора.
PROCESSOR_LEVEL=6 - уровень (номер модели) процессора.
PROCESSOR_REVISION=0801 - версия процессора.
ProgramFiles=C:\Program Files - путь к папке "Program Files"
PROMPT=PG - формат приглашения командной строки $P - путь для текущего каталога $G - знак ">".
SystemDrive=C: - буква системного диска.
SystemRoot=C:\WINDOWS - каталог ОС Windows.
Значение некоторых переменных по команде SET не выдаются. В основном, это переменные, принимаемые значения которых динамически изменяются:
%CD% - Принимает значение строки текущего каталога.
%DATE% - Принимает значение текущей даты.
%TIME% - Принимает значение текущего времени.
%RANDOM% - Принимает значение случайного десятичного числа в диапазоне 1 -32767.
%ERRORLEVEL% - Принимает текущее значение кода завершения задачи ERRORLEVEL
%CMDEXTVERSION% - Принимает значение версии командного процессора CMD.EXE для расширенной обработки команд.
%CMDCMDLINE% - Принимает значение строки, которая вызвала командный процессор.

Для просмотра действующего значения какой-либо переменной обычно используется команда:

ECHO %переменная%:
ECHO %CD% - отобразить имя текущего каталога;
ECHO %TIME% - отобразить текущее время;
ECHO %ERRORLEVEL% - отобразить результат выполнения предыдущей команды.

Значения, принимаемые переменными окружения, могут быть расширены с помощью специального признака - символа **" ~ "**, что получить частичное значение (расширение переменной), или изменить его заменой какой-либо части. Примеры использования расширений переменных рассмотрены ниже.

3. **Передача параметров командному файлу.**

Очень полезной особенностью работы с командными файлами является возможность передавать параметры командной строки и использовать их значения в операциях внутри самого командного файла.

BAT-файл параметр1 параметр2 ... параметрN.

В самом командном файле первый параметр будет доступен как переменная **%1**, второй - **%2** и т.п. Путь и имя самого командного файла доступно как переменная **%0**. Для примера создадим командный файл, задачей которого будет выдача на экран значений введенных при его запуске параметров командной строки. Для вывода текста на экран используется команда **ECHO текст**, однако если "текст" заменить на %0, - то будет выдано имя командного файла, %1 - первый аргумент, заданный в строке запуска, %2 - второй и т.д.

Создаем, например, командный файл params.bat следующего содержания:

echo off echo Это командный файл %0
echo Первый параметр=%1
echo Второй параметр=%2
echo Третий параметр = %3

и запускаем его на выполнение следующей командой:

params.bat FIRST second "two words"

параметры содержащие пробелы, нужно заключать в двойные кавычки. В первой строке командного файла используется команда "echo off" для того, чтобы обрабатываемые командным процессором строки не выдавались на экран.

Для проверки наличия каких-либо входных параметров, передаваемых командному файлу, можно проверить, является ли значение переменной %1 пустым:

if "%1" EQU "" goto error
....
...
:error

4. Переходы и метки

В командных файлах можно использовать команды условного перехода, меняющие логику их работы в зависимости от выполнения определенных условий. Для иллюстрации приемов использования условных переходов создадим командный файл, целью которого будет присвоение заранее определенной буквы для съемных носителей, в качестве которых будут использоваться флэш-диски. Условия таковы - есть 2 флэш-диска, один из которых должен быть виден в проводнике как диск X: а второй - как диск Y: независимо от того, в какой порт USB они подключены и какие буквы присвоены им операционной системой. Будем считать, что реальные диски могут быть подключены как F: или G: Опознавание дисков будем выполнять по наличию файла с определенным именем (лучше такой файл сделать скрытым в корневом каталоге и назвать его как-нибудь необычно):

Flashd1.let - на первом диске
Flashd2.let - на втором диске

Таким образом, задача командного файла заключается в том, чтобы проверить наличие на сменных дисках **F:** и **G:** файлов **Flashd1.let** или **Flashd2.let** и, в зависимости от того, какой из них присутствует, присвоить диску букву **X:** или **Y:**

Для поиска файла на диске воспользуемся командой **IF EXIST: IF EXIST имя_файла команда.**

В качестве команды проще всего воспользоваться **SUBST**, сопоставляющей имя диска и каталог.

SUBST X: C: - - создать виртуальный диск X:, содержимым которого будет корневой каталог диска C:. Для решения задачи, создаем командный файл, например **setletter.bat**, следующего содержания:

```
@ECHO OFF
IF EXIST G:\flashd1.let SUBST X: G:\
IF EXIST F:\flashd1.let SUBST X: F:\
IF EXIST G:\flashd2.let SUBST Y: G:\
IF EXIST F:\flashd2.let SUBST Y: F:\
```

После выполнения этого командного файла у вас появятся диски X: и Y:. Однако, если такой файл выполнить повторно, команда SUBST выдаст сообщение об ошибке - ведь диски X: и Y: уже существуют. Поэтому, желательно обойти выполнение SUBST, если виртуальные диски X: и Y: уже созданы, или удалять их, используя SUBST с параметром -d перед подключением. Попробуйте изменить командный файл setletter.bat с использованием команды перехода **GOTO**, осуществляющей передачу управления строке пакетного файла на указанную метку:

GOTO метка

В качестве метки используется строка символов, начинающаяся с двоеточия. Сделаем изменения в нашем командном файле, чтобы не возникало сообщений об ошибке:

```
@ECHO OFF
REM если не существует X: - то перейдем на метку SETX
IF NOT EXIST X:\ GOTO SETX
REM если существует X: - перейдем на проверку наличия Y:
GOTO TESTY
:SETX
IF EXIST G:\flashd1.let SUBST X: G:\
IF EXIST F:\flashd1.let SUBST X: F:\
:TESTY
REM если Y: существует - завершим командный файл.
IF EXIST Y:\ GOTO EXIT
IF EXIST G:\flashd2.let SUBST Y: G:\
IF EXIST F:\flashd2.let SUBST Y: F:\
REM выход из командного файла
:EXIT
```

При выполнении измененного таким образом командного файла, сообщение об ошибке при выполнении SUBST исчезнет.

Одним из важнейших приемов при написании сложных командных файлов является анализ успешности выполнения конкретной команды или программы. Признаки ошибок при выполнении команд можно отслеживать, анализируя специальную переменную **ERRORLEVEL**, значение которой формируется при выполнении большинства программ. Обычно ERRORLEVEL равно нулю, если программа завершилась без ошибок и единице - при возникновении ошибки. Могут быть и другие значения, если они предусмотрены в выполняемой программе.

В качестве команды в строке командного файла можно использовать также командный файл. Причем, для передачи с возвратом обратно к точке выполнения вызывающего командного файла используется команда **CALL**. Попробуйте создать командный файл **test.bat**, следующего содержания:

```
@ECHO OFF
ECHO Вызов 1.bat
CALL 1.bat
ECHO Возврат.
```

В этом же каталоге, создайте второй файл под именем **1.bat**, содержащий команду **PAUSE** , приостанавливающую выполнение командного файла до нажатия любой клавиши.

@ECHO OFF
pause
При выполнении командного файла **test.bat** будет выдано на экран сообщение
Вызов 1.bat
и управление получит командный файл 1.bat с одной единственной командой pause. После нажатия клавиши на клавиатуре управление будет возвращено вызвавшему командному файлу на строку "ECHO Возврат." и на экран будет выдано
Возврат.
Если же в файле test.bat убрать CALL, оставив "1.bat", то возврат выполняться не будет. Вызываемый командный файл может создавать переменные и присваивать им определенные значения, которые будут доступны для обработки в вызывающем файле. Попробуйте изменить файл test.bat на следующее содержимое:
@ECHO OFF
ECHO Вызов 1.bat
CALL 1.bat
ECHO Получено из файла %MYFILE% значение MYNUMBER=%MYNUMBER%,

а в файле 1.bat на следующее
@ECHO OFF
SET MYFILE="Very good 1.bat"
SET MYNUMBER=99
Используя передачу управления командному файлу, можно организовать его зацикливание. Добавите в конец файла test.bat строку:
CALL test.bat
Выйти из зацикливания командного файла можно по нажатию комбинации **CTRL-Break**. Возможно использование команды CALL для вызова процедуры внутри командного файла. В этом случае в качестве аргумента используется не имя внешнего файла, а метка:
call :proc1
....
:proc1
....
exit
....

5. Примеры командных файлов

Использование утилит командной строки и командных файлов нередко позволяют решить многие проблемы связанные с повседневной эксплуатацией компьютерной техники. Большинство системных администраторов и грамотных пользователей продолжают ими пользоваться, несмотря на то, что в Windows проявилось новое, более мощное и современное средство управления системой - WMI (Windows Management Instrumentation). Очевидно, не в последнюю очередь, это обусловлено простотой реализации и, тем не менее, - достаточной эффективностью использования командных файлов. Ниже приведены простые примеры с комментариями, которые демонстрируют некоторые возможности и способы применения .cmd и .bat .

Своя команда для создания новых файлов

В составе операционной системы Windows нет специальной команды для создания нового файла, но без нее можно легко обойтись несколькими способами:

Копирование с клавиатуры в файл

COPY CON myfile.txt

При выполнении этой команды данные с клавиатуры (стандартное устройство CON - консоль) будут заноситься в файл myfile.txt. Нажатие клавиши F6 или комбинации CTRL-Z завершит вывод.

Перенаправление вывода

ECHO 1 > myfile.txt

При выполнении этой команды будет создан файл myfile.txt, содержащий символ "1".

Комбинация перенаправления ввода и перенаправления вывода:

COPY CON > myfile.txt < xyz

При выполнении этой команды, как и в первом случае, используется копирование с консоли в файл, но вместо ручного ввода данных с клавиатуры используется ввод с несуществующего файла xyz. Система выдаст сообщение, о том, что такого устройства или файла не существует, но пустой файл myfile.txt будет успешно создан.
Еще проще использовать команду копирования из фиктивного устройства **nul** в файл. Использование устройства nul позволяет обойти стандартные операции ввода-вывода, которые для него реально не выполняются:

COPY NUL myfile.txt

При работе в командной строке часто приходится создавать новые пустые файлы, поэтому, стоит подготовить свой командный файл (например, с именем nf.bat), а имя нового создаваемого файла передавать ему в качестве параметра при запуске.
Содержимое файла:

```
@echo off
REM Создание пустого файла, имя которого задано в строке запуска
if "%1" EQU "" goto error
copy nul %1
goto exit
:error
ECHO ОШИБКА: Необходимо задать имя нового файла!
:exit
```

Для простоты использования, поместите этот командный файл в системный каталог (например, в C:\windows\system32) или любой другой, существующий в путях поиска, задаваемых значением переменной PATH). Теперь, в командной строке, находясь в любом каталоге можно одной командой создавать пустые файлы.

Командная строка:

nf.bat myfile.txt - создать файл с именем myfile.txt в текущем каталоге.

nf.bat C:\myfile.txt - создать файл в корневом каталоге диска C:

nf.bat "%USERPROFILE%\myfile.txt" - создать файл в каталоге профиля текущего пользователя.

Расширение командного файла (.bat) можно не набирать и команда еще больше упрощается: **nf myfile.txt**

В тексте командного файла присутствует проверка, задано ли имя создаваемого файла в командной строке (if "%1%" EQU "" goto error), и если не задано - выводится сообщение об ошибке и командный файл завершает свою работу. Добавите в этот командный файл проверку на существование файла с именем, указанным в командной строке.

6. Присвоение съемному диску одной и той же буквы

Задача заключается в том, чтобы съемный USB диск (флэш диск) был доступен всегда под одной и той же буквой, независимо от того, на каком компьютере он используется и как он подключен. Для ее решения воспользуемся уже упоминаемой выше командой **SUBST**, но реализуем присвоение новой буквы диску с использованием подстановочного значения переменной **%0**, создаваемой системой при каждом запуске командного файла.
Выберем для съемного диска желаемую букву, например - X.

Некоторые из переменных окружения, в том числе и переменная **%0** принимающая значение пути и имени выполняющегося командного файла, позволяют при определенной модификации с использованием специального признака - символа " ~ " получить ее частичное значение (расширение переменной). Например, не полный путь файла, а только его имя, или каталог расположения, или букву диска, с которого он был запущен или еще около десятка различных элементов, связанных с подстановочными значениями переменной %0. Имя диска, с которого был запущен командный файл доступно как переменная **%~d0**. Теперь создаем командный файл следующего содержания:

@echo off
subst X: %~d0

что означает - создать виртуальный диск X:, которому сопоставлен логический диск, являющийся частью пути данного командного файла. Если такой файл записать на флэшку, и выбрать присваиваемую букву диска поближе к концу алфавита (чтобы не оказалась занятой другим реальным дисковым устройством) то после его запуска, в системе будет создаваться новый диск всегда под одной и той же буквой.

Дополнительное представление о подстановочных значениях переменной %0 можно получить из командного файла следующего содержания:

```
@echo off
ECHO ОБРАБАТЫВАЕТСЯ ФАЙЛ - %0
ECHO Дата/время создания/изменения командного файла - %~t0
ECHO Путь командного файла - "%~f0"
ECHO Диск командного файла - %~d0
ECHO Каталог командного файла - "%~p0"
ECHO Имя командного файла - %~n0
ECHO Расширение командного файла - %~x0
ECHO Короткое имя и расширение - %~s0
ECHO Атрибуты командного файла - %~a0
ECHO Размер командного файла - %~z0
```

7. Создание архива, имя которого содержит дату и время

Решим следующую задачу - нужно создать архив файлов, находящихся в каталоге C:\Program Files\FAR. Имя архивного файла должно состоять из текущего времени (часы.минуты.секунды - ЧЧ.ММ.СС.rar), и помещен он должен в новый каталог, имя которого должно состоять из текущей даты (день.месяц.год - ДД.ММ.ГГГГ). Для архивирования будем использовать архиватор RAR. Формат запуска для создания архива: **RAR a -r < путь и имя архива > < Путь и имя архивируемых данных >**

a - команда создания архива.

-r - ключ, определяющий архивирование подкаталогов (т.к. в исходной папке есть подкаталоги).

Таким образом, для решения задачи нужно правильно создать имена и пути для RAR. Для чего воспользуемся следующими исходными данными:

- в командных файлах можно получить доступ к текущей дате и текущему времени - переменные %DATE% и %TIME%;
- в командных файлах можно создавать временные переменные с помощью команды SET;

Значение временных переменных может быть сформировано на основе %DATE% и %TIME% путем пропуска и (или) замещения их частей с помощью специальной конструкции с использованием символа **~** и числового значения, определяющего группу символов из данных текущего значения переменной.

Дата, получаемая из переменной %DATE% при стандартных настройках региональных установок Windows 2000 выглядит следующим образом:

Пн 21.01.2014 - День недели (2 символа)-Пробел(1 символ)-дата(10 символов) - всего 13 символов. В Windows XP/Vista/7 день недели отсутствует, что несколько упрощает структуру даты. Для создания нового каталога в командной строке используется команда **MD имя каталога**.

Имя каталога нужно получить из текущей даты. Создаем в памяти временную переменную VDATE и присваиваем ей значение переменной окружения DATE, без первых 3-х символов (Пн и пробел) - 20.01.2014:

set VDATE=%date:~3%

В версиях Windows, где в значении принимаемом переменной DATE, отсутствует день недели (3 символа - "Пн "), значение VDATE получится не тем, что требуется. Чтобы не анализировать признаки наличия данного кода, можно воспользоваться и другим вариантом - не пропустить первые 3 символа (~3) от начала строки переменной DATE, а взять 10 символов от конца строки, указав число 10 со знаком "минус" - будет тот же результат - 20.01.2010

set VDATE=%date:~-10%

Создаем каталог на диске C:, имя которого = текущая дата из переменной VDATE: **MD C:\%VDATE%**

После выполнения этой команды на диске C: будет создан каталог с именем 20.01.2014. Можно обойтись без лишних операторов, связанных с формированием значения переменной VDATE, которую я использовал для упрощения понимания структуры создаваемого имени каталога:

MD %DATE:~-10% - создать каталог, имя которого будет представлено в виде текущей даты ДД.ММ.ГГГГ

Время, получаемое из переменной %TIME% выглядит так : **14:30:59.93** - Часы, минуты, секунды, сотые доли секунды.

Сотые доли - это в имени файла архива, пожалуй, лишнее. Создаем временную переменную VTIME и присваиваем ей текущее время без последних 3-х символов, т.е пропускаем 0 символов от начала и отсекаем 3 символа от конца. Количество пропущенных и отсекаемых символов разделяются запятой:

set VTIME=%time:~0,-3%

Теперь VTIME = 14:30:59, но знак двоеточия (:) в имени файла использовать нельзя, это специальный символ, использующийся в именах устройств (диск C:\). Поэтому, его придется заменить его на любой другой символ, допустимый в имени файла, например, точку. Для замены символов используется знак " = "

set VTIME=%VTIME::=.% - заменить в переменной VTIME символ двоеточия на символ точки.

Переменная VTIME примет значение 14.30.59

Запустим архиватор:

rar.exe a -r C:\%VDATE%\%VTIME%.rar "C:\Program files\far*.*"

Теперь можно создать командный файл с содержимым:

```
set VDATE=%date:~-10%
md c:\%VDATE%
set VTIME=%time:~0,-3%
set VTIME=%VTIME::=.%
rar.exe a -r C:\%VDATE%\%VTIME%.rar "C:\Program files\far\*.*"
```

Такой командный файл можно выполнять через автозагрузку, или как часть скрипта, при входе пользователя в домен, либо с помощью планировщика в заданное время, и у вас всегда будут в наличии упорядоченные по времени архивы критических данных.

1.8. Создания архива каталога "Мои Документы"

Этот командный файл создает архивы содержимого папки "Мои Документы" пользователей Win2K/XP, размещая их в каталоги C:\ARHIV\Мои документы\Имя пользователя\Дата\время. При этом используются переменные окружения USERPROFILE, USERNAME, WINDIR. В файле используется команда rem, дающая некоторые комментарии:

```
@echo off
    rem Задается переменная FROM - откуда брать данные для архивирования
  set FROM=%USERPROFILE%\Мои Документы
   rem Задается переменная TO - куда помещать архивы
   set TO=C:\arhiv\Мои документы\%USERNAME%
   rem Создадим каталог TO
   md "%TO%\"
   rem Сформируем имя подкаталога из текущей даты
   set VDATE=%date:~-10%
     rem Сформируем имя файла архива из текущего времени - 12:00:00.99
  rem отбросим сотые доли секунды и заменим символ : на символ . Результат - 12.00.00
  set vtime=%TIME:~0,-3%
   set vtime=%vtime::=.%
   rem Создадим подкаталог для файла архива
   md "%TO%\%VDATE%"
   rem Команда для архивирования. Ключ -r нужен для архивирования с вложенными папками
  rem вариант для архиватора ARJ : arj.exe a -r "%TO%\%VDATE%\%VTIME%.arj" "%FROM%\*.*"
  rem При использовании архиватора RAR:
  rar.exe a -r "%TO%\%VDATE%\%VTIME%.rar" "%FROM%\*.*"
```

Если возникают проблемы связанные с неверной кодировкой символов русского алфавита в именах файлов и каталогов, воспользуйтесь командой **CHCP** для смены кодовой страницы **chcp 866** - установить кодовую страницу 866 (DOS-кодировка);
chcp 1251 - установить кодовую страницу 1251 (Windows-кодировка).

Этот командный файл можно значительно сократить, убрав ненужные переменные VTIME и VDATE, которые в данном примере, используются лишь для того , чтобы скрипт имел более наглядный и простой для понимания вид.

В операционных системах Windows XP/Vista/7 формат даты по умолчанию не содержит название дня недели. Если есть необходимость получить это значение без изменения настроек системы и использования дополнительного программного обеспечения, можно воспользоваться сценарием Hindows Script Host (WSH).

- создаем файл сценария для получения названия дня недели, пусть с именем weekday.vbs, и содержащим строку вывода на экран результата выполнения функции WeekDayName **WScript.Echo WeekDayName(Weekday(Now), True);**
- выполняем скрипт WSH с использованием консольной версии программы обработки сценариев cscript.exe и подавлением лишних сообщений (ключ //nologo) **cscript //nologo weekday.vbs**

Пример командного файла для получения названия дня недели с использованием функции WeekDayName :

```
ECHO OFF
echo WScript.Echo WeekDayName(Weekday(Now), True) > weekday.vbs
for /f "Tokens=1*" %%i in ('cscript /nologo weekday.vbs') DO set DayName=%%i
echo %DayName%
```

REM Дальше можно использовать переменную DayName, а файл weekday.vbs – удалить
REM ERASE dayname.vbs
REM ...

9. Выполнение команд по расписанию

В операционных системах WINDOWS 2000/XP и старше существует утилита командной строки **AT.EXE**, позволяющая управлять задачами для **планировщика заданий** Windows, и таким образом, выполнить команду или пакетный файл в указанное время на локальном или удаленном компьютере. Естественно, для успешного функционирования команды **AT** необходимо, чтобы была запущена системная служба **Планировщик заданий** (обычно она существует и запускается автоматически при стандартной установке системы).

Примеры команды:

AT [\\имя_компьютера] [[код] [/DELETE] | /DELETE [/YES]]

AT [\\имя_компьютера] время [/INTERACTIVE] [/EVERY:день[,...] | /NEXT:день[,...]] "команда"

где:

\\имя_компьютера - имя удаленного компьютера. Если этот параметр опущен, задача относится к локальному компьютеру;

код - порядковый номер запланированной задачи. Указывается если нужно отменить уже запланированную задачу с помощью ключа **/delete**;

/delete - отменить запланированную задачу. Если код задачи опущен, отменяются все задачи, запланированные для указанного компьютера;

/yes - не будет запроса на подтверждение при отмене всех запланированных задач;

время - время запуска команды;

/interactive - интерактивный режим, разрешение взаимодействия задачи с пользователем. Задачи, запущенные без этого ключа невидимы для пользователя компьютера;

/every:день[,...] Запуск задачи осуществляется по указанным дням недели или месяца. Если дата опущена, используется текущий день месяца;

/next:день[,...] Задача будет запущена в следующий указанный день недели (например в следующий четверг). Если дата опущена, используется текущий день месяца;

"команда" - Команда или имя командного файла.

Примеры использования:

- Просмотр списка запланированных задач: **AT**;
- Удаление уже спланированных задач: **AT 3 /DELETE** - удаление задачи с номером 3;

AT /DELETE /YES - удаление всех задач без запроса подтверждения;

- Создание интерактивных задач

AT \\SERVER 15:21 /interactive notepad.exe - на компьютере SERVER в 15:21 запустить видимое для пользователя приложение "Блокнот" (notepad.exe)$

AT 15:30 /interactive regedit.exe - в 15:30 запустить видимый редактор реестра на своем компьютере.

- Аналог "будильника" - всплывающие окна с текстом, напоминающие о необходимости каких-либо действий. Для посылки сообщения удаленному пользователю используется утилита **NET.EXE** в режиме отправки сообщения **SEND**. На компьютерах должна быть запущена служба сообщений, иначе NET SEND не будет работать:

AT 17:30 net.exe send COMP Пора домой - в 17:30 отправить сообщение "Пора домой" пользователю компьютера COMP;

AT \\PROXY 15:30 net.exe send COMP2 Test Message - создать задание на компьютере PROXY, чтобы в 15:30 им было отправлено сообщение "Test Message" на компьютер COMP2;

AT 15:45 net.exe send имя своего компьютера Task Scheduler test - в 15:45 на своем компьютере показать сообщение "Task Scheduler test".

Для доступа к удаленному компьютеру и создания заданий, пользователь, выполняющий команду **AT** должен обладать соответствующими правами по отношению к удаленной системе.

Создаваемые командой **AT** задачи доступны для обработки в среде пользователя с помощью оснастки "Назначенные задания" Windows: **Пуск -> Панель управления -> Назначенные задания** - здесь можно просматривать, изменять и удалять созданные командой **AT** задачи.

10. Остановка и запуск системных служб

Для остановки и запуска служб из командной строки, в любой версии Windows, можно воспользоваться командой NET.EXE:

NET.EXE STOP < имя службы >

NET.EXE START < имя службы >

В качестве параметра команды можно использование как короткое, так и полное имя службы ("Dnscache" - короткое, "DNS-клиент" - полное имя службы). Имя службы, содержащее пробелы заключается в двойные кавычки. Пример перезапуска службы "DNS-клиент":

net stop "DNS-клиент"

net start "DNS-клиент"

То же, с использованием короткого имени:

net stop Dnscache

net start Dnscache

Полное имя службы можно скопировать из: **"Панель управления" -> "Администрирование" -> "Службы" -> Имя службы - >"Свойства" -> "Выводимое имя"**. То же самое, но в режиме командной строки: **"Пуск" - "Выполнить" - services.msc**.

Для управления службами гораздо удобнее воспользоваться утилитой PsService.exe из ***утилит PsTools***. Утилита не требует установки и работает в любой OS Windows. Кроме запуска и остановки, позволяет выполнить поиск конкретной службы на компьютерах локальной сети, опросить состояние и конфигурацию службы, изменить тип запуска, приостановить службу, продолжить, перезапустить. Для работы с системными службами в Windows XP и старше, можно использовать утилиту **sc.exe**, позволяющую не только остановить/запустить службу, но и опросить ее состояние, параметры запуска и функционирования, изменить конфигурацию, а также работать не только с системными службами, но и с драйверами. При наличии соответствующих прав, можно управлять службами не только на локальной, но и на удаленной машине. Примеры:
sc.exe stop DNSCache - остановить службу DNSCache на локальном компьютере;
sc \\192.168.0.1 query DNSCache - опросить состояние службы DNSCache на компьютере с IP-адресом 192.168.0.1;
sc \\COMP start DNSCache запустить службу DNSCache на компьютере COMP
Подсказку по работе с утилитой можно получить, введя:
sc /?

1.11. Диалог с пользователем

Для диалога с пользователем можно использовать команду:

SET /P имя переменной = текст

при выполнении которой, на экран выдается текстовое сообщение < текст > и ожидается ввод ответа. Пример - выполним запрос пароля и присвоим его значение переменной "pset":

set /p pset="Enter password - "

echo Password is - %pset%

Недостатком данного способа является невозможность продолжения выполнения командного файла при отсутствии ответа пользователя, поэтому очень часто вместо **set** используются сторонние программы. В составе операционных систем семейства Microsoft Windows имеется утилита командной строки **CHOICE** позволяющая довольно просто реализовать диалог с пользователем и проанализировать введенные им данные, однако в разных версия ОС утилита может присутствовать в стандартной поставке (Windows 7) или входить в наборы дополнительных программных инструментов (Resource Kit Windows XP). Простейшая версия - **CHOICE.COM**, работающая во всех ОС семейства Windows.

CHOICE выдает пользователю текстовое сообщение и ожидает выбора одного из заданных вариантов ответа (нажатия клавиш на клавиатуре). По результатам выбора формируется переменная ERRORLEVEL, значение которой равно порядковому номеру выбора. По умолчанию вариантов выбора два - Y или N. Если ответ равен Y - то ERRORLEVEL=1, если N - то ERRORLEVEL=2. Можно использовать более 2-х вариантов выбора и есть возможность задать выбор по умолчанию, когда пользователь за определенное время не нажал ни одной клавиши. Формат командной строки:

CHOICE [/C[:]choices] [/N] [/S] [/T[:]c,nn] [text]

/C[:]choices - определяет допустимые варианты выбора. Если не задано – Y/N

/N - не выдавать варианты выбора;

/S - строчные и заглавные буквы отличаются.

/T[:]c,nn - Выбор по умолчанию равен "c" через "nn" секунд

text - Строка текста выводимая в качестве запроса

Создадим командный файл, демонстрирующий использование CHOICE. Он будет реагировать на нажатие клавиш "1","2",3" и "0" . При нажатии "0" выполняется завершение, а при нажатии остальных - сообщение пользователю. Если в течении 10 секунд ничего не нажато – завершение:

```
@ECHO OFF
:CHOICE
CHOICE /C:1230 /T:0,10 Ваш вариант
IF %ERRORLEVEL% EQU 4 GOTO EXIT
echo Ваш выбор=%ERRORLEVEL%
GOTO CHOICE
:EXIT
```

Теперь, используя CHOICE можно создавать командные файлы, логика работы которых может определяться пользователем.

12. Определение доступности IP-адреса

Для проверки доступности сетевого узла используется стандартная утилита ping.exe. Утилита выполняет отправку ICMP-пакета на проверяемый узел (эхо-запрос) и ожидает ответный пакет (эхо-ответ). Результат проверки никак не отражается в переменной ERRORLEVEL и может быть получен только в данных стандартного вывода ping. Ненулевое значение ERRORLEVEL утилита ping.exe формирует только в том случае, если заданы ошибочные параметры командной строки. Иными словами, в некоторых случаях, нужный результат выполнения определенной команды нельзя определить по значению переменной ERRORLEVEL, и приходится анализировать, например, результат текстового вывода. Если внимательно посмотреть на сообщения программы ping.exe при опросе доступного и недоступного узла, то можно заметить, что они значительно отличаются :

ping 456.0.0.1 - ping на несуществующий адрес.

Ответ на такую команду может отличаться от конкретной версии утилиты, и может быть приблизительно таким:

При проверке связи не удалось обнаружить узел 456.0.0.1. Проверьте имя узла и повторите попытку.

ping yandex.ru - ping на адрес узла yandex.ru.

Ответ на ping доступного узла:

Обмен пакетами с yandex.ru [87.250.250.11] по 32 байт:
Ответ от 87.250.250.11: число байт=32 время=10мс TTL=55

Таким образом, для решения задачи определения доступности узла в командном файле, достаточно проанализировать характерные слова в выводе ping.exe при успешном ответе. Наиболее характерно в данном случае наличие слова **TTL**. Оно никогда не встречается при возникновении ошибки и состоит всего лишь из символов английского алфавита. Для поиска "TTL" в результатах ping.exe удобнее всего объединить ее выполнение в цепочку с командой

поиска строки символов **FIND.EXE** (конвейер ping и find). Справку по использованию можно получить командой **find /?**

Поиск текстовой строки в одном или нескольких файлах:

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]] "строка" [[диск:][путь]имя_файла[...]]

где:

/V - вывод всех строк, НЕ содержащих заданную строку;

/C - вывод только общего числа строк, содержащих заданную строку;

/N - вывод номеров отображаемых строк;

/OFF[LINE] - не пропускать файлы с установленным атрибутом **"Автономный";**

/I - поиск без учета регистра символов;

"строка"- искомая строка;

[диск:][путь]имя_файла - один или несколько файлов, в которых выполняется поиск. Если путь не задан, поиск выполняется в тексте, введенном с клавиатуры либо переданном по конвейеру другой командой.

Как видно из справки, **find.exe** можно использовать для поиска нужной строки символов в тексте, переданном по конвейеру командой ping.exe. Если текст найден, значение переменной ERRORLEVEL будет равно **0.**

```
ping -n 1 COMPUTER | find /I "TTL" > nul
if %ERRORLEVEL%==0 goto LIVE
ECHO computer не доступен
подпрограмма обработки недоступного состояния
...
Exit
```

:LIVE - начало подпрограммы обработки состояния доступности узла

...

В конвейер добавлена команда перенаправления стандартного вывода на фиктивное устройство **nul**, т.е. подавление вывода. Ключ **-n 1** задает однократный опрос узла COMPUTER для ping.exe.

13. Определение текущей версии Windows

Для определения версии операционной системы в процессе выполнения командного файла, можно воспользоваться поиском определенных фрагментов текста в результатах выполнения команд, отображающих сведения о системе. Например, во всех операционных системах семейства Windows (и даже в DOS) существует специальная команда **VER**, предназначенная для отображения сведений о версии ОС. В результате выполнения команды, например, в среде Windows XP, отображается текст:

Microsoft Windows XP [Версия 5.1.2600]

В среде Windows 7, текст отличается:

Microsoft Windows [Version 6.1.7600]

Таким образом, результат выполнения команды VER в среде разных версий Windows, всегда содержит определенный текст, характерный только для данной ОС, и задача определения версии решается довольно просто:

```
@echo off
set curr_OS=
REM
ver | find /i "5.0"
if %errorlevel% == 0 set curr_OS=Windows 2000
REM
ver | find /i "5.1"
if %errorlevel% == 0 set curr_OS=Windows XP
REM
ver | find /i "5.2.3"
if %errorlevel% == 0 set curr_OS=Windows Server 2003
```

```
REM
ver|find /i "6.0"
if %errorlevel% == 0 set curr_OS=Windows Vista
REM
ver | find /i "6.1">nul
if %errorlevel% == 0 set curr_OS=Windows 7
REM
if "%curr_OS%"=="" set curr_OS=Unknown
echo Текущая версия ОС - %curr_OS%
```

Можно также воспользоваться более информативным выводом команды **NET CONFIG WORKSTATION**. При выполнении в среде Windows XP вывод команды представляет собой следующий текст:

```
Имя компьютера                  \\COMP1
Полное имя компьютера            COMP1.Mydomain
Имя пользователя                USER2
Активная рабочая станция на
     NetbiosSmb (000000000000)
     NetBT_Tcpip_{F53DEAF8-0AF5-4875-B565-8ED55C594769} (000D87009D28)
Версия программы                 Windows 2002
Домен рабочей станции             Mydomain
DNS-имя домена рабочей станции    Mydomain
Домен входа                      Mydomain
Интервал ожидания открытия COM-порта (с)    0
Отсчет передачи COM-порта (байт)           16
Таймаут передачи COM-порта (мс)            250
Команда выполнена успешно.
```

Для среды Windows 7 результат выполнения команды выглядит так:

```
Имя компьютера                  \\COMP1
Полное имя компьютера            COMP1.Mydomain
Имя пользователя                user2
Активная рабочая станция на
     NetBT_Tcpip_{F53DEAF8-0AF5-4875-B565-8ED55C594769} (000D87009D28)
Версия программы                 Windows 7 Professional
Домен рабочей станции             Mydomain
Домен входа                      Mydomain
Интервал ожидания открытия COM-порта (с)    0
Отсчет передачи COM-порта (байт)           16
Таймаут передачи COM-порта (мс)            250
Команда выполнена успешно.
```

Строка **Версия программы . . .** тоже может быть использована для определения версии Windows, в среде которой выполняется командный файл. Кроме того, в результатах выполнения команды **NET CONFIG WORKSTATION** для серверных версий Windows всегда присутствует слово **Server**.

```
@echo off
set curr_OS=
REM
net config workstation | find /i "Windows 2000"
if %errorlevel% == 0 set curr_OS=Windows 2000
REM
```

net config workstation | find /i "Windows 2002"
if %errorlevel% == 0 set curr_OS=Windows XP
REM
net config workstation | find /i "Server 2003"
if %errorlevel% == 0 set curr_OS=Windows Server 2003
REM
net config workstation|find /i "Windows Vista"
if %errorlevel% == 0 set curr_OS=Windows Vista
REM
net config workstation | find /i "Windows 7">nul
if %errorlevel% == 0 set curr_OS=Windows 7
REM Плюс поиск по "Professional"
net config workstation | find /i "Версия программы" | find "Professional"
if errorlevel 0 if not errorlevel 1 set curr_OS=Windows 7 PRO
REM Если версия неизвестна:
if "%curr_OS%"=="" set curr_OS=Unknown
echo %curr_OS%

14. Выключение компьютеров по списку, созданному на основе сетевого окружения

Выключение производится утилитой PsShutdown.exe. Сначала создается файл со списком компьютеров на основе сетевого окружения, а затем выполняется их поочередное выключение, при условии, что компьютер не свой (иначе он может выключиться до окончания выполнения командного файла). Содержимое файла:

rem @echo off
REM Здесь нужно задать
REM имя домена или рабочей группы для которых строится список машин для выключения:
set MyDomain=имя домена
REM
REM Создадим текстовый файл comps.txt со списком компьютеров с помощью NET VIEW
net view /DOMAIN:%MyDomain% > comps.txt
REM
REM FOR /F "параметры" - использование данных из файла
REM eol=K - не использовать строки, начинающиеся с "K" - "Команда выполнена успешно"
REM skip=4 - пропустить первые 4 строки в файле
REM tokens=1 - брать для обработки 1-е слово в строке
FOR /F "eol=K skip=4 tokens=1 " %%i in (comps.txt) do (
REM Свой компьютер выключать не будем
REM Если имя компьютера не равно COMPUTERNAME – выключаем
IF /I %%i NEQ %COMPUTERNAME% psshutdown -k -t 0 %%i
)
Вам нужно только подредактировать строку:
set MyDomain=
указав имя домена и, при необходимости, добавить параметры **-u -p** для **psshutdown.exe** .

В реальной жизни из списка выключаемых компьютеров нужно исключить несколько штук, для чего удобно использовать команду **FIND** в цепочке с **net.exe** в скрипте формирования списка на основе сетевого окружения. Данная команда используется для поиска строк в текстовом файле по шаблону. Ключ **/V** используется для поиска строк **не** совпадающих с шаблоном. Для выключения компьютеров, исключая server1...server4 удобно использовать такой вариант:

net view | find "\\" | find /v "сервер1" | find /v "сервер2" | find /v "сервер3" | find /v "сервер4" > comps.txt
FOR /F "tokens=1 " %%i in (comps.txt) do shutdown.exe -f -s -m %%i

15. Работа с дисками, файлами и каталогами

Задача - определить буквы дисков, присутствующих в системе и записать результат в файл с именем tstdsk.txt текущего каталога. Можно воспользоваться выполнением команды **IF EXIST** в цикле FOR для набора из букв латинского алфавита, т.е для каждой буквы диска проверить наличие корневого каталога командой:

IF EXIST буква диска:

Сначала создаем пустой файл:

copy nul tstdsk.txt

Это действие необязательно, если файла не существует, но в противном случае, результаты будут дописываться в конец файла, и если в нем уже был список дисков от предыдущего исполнения командного файла, то он удвоится. Команда **copy nul tstdsk.txt** для существующего файла установит нулевой размер данных, т.е. сделает его пустым. Окончательно, командный файл будет выглядеть следующим образом:

```
copy nul tstdsk.txt
for %%i in (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z) DO (
if exist %%i:\ echo Disk %%i: exist >> tstdsk.lst
)
```

Для обработки файлов определенного типа, например любых с расширением **.tmp** используется маска - ***.tmp** . Так, для удаления всех файлов *.tmp из каталога C:\TEMP можно воспользоваться командой ERASE (или DEL)

ERASE C:\TEMP*.TMP
DEL /Q C:\TEMP*.TMP

В масках файлов и каталогов возможно использование частичных имен:

ERASE C:\TEMP\A*.TMP - удалить все файлы с расширением .TMP , имя которых начинается с символа "A";

DIR *u*.* - выдать список всех файлов и подкаталогов текущего каталога, в имени которых содержится символ "u";

DIR C:*t.* - выдать список всех файлов и каталогов в корне диска C: , имя которых заканчивается символом "t".

Задача - получить список всех каталогов с подкаталогами на логическом диске и записать результат в текстовый файл. Для рекурсивной обработки каталогов диска будем использовать команду **FOR /R**:

FOR /R [[диск:]путь] %переменная IN (набор) DO команда [параметры].

Ключ **/R** означает выполнение команды для каталога **[диск:]путь**. Если в команде путь не задан, то обработка выполняется для текущего каталога. Простой пример удаления файлов с расширением .tmp из каталога C:\TEMP:

FOR /R C:\temp\ %%i IN (*.tmp) DO del %%i

При выполнении команды возможно использование подстановочных значений переменной цикла для получения имен дисков, папок, файлов и их характеристик. Полный список возможных значений в случае использования переменной с именем **i:**

%%~i - из переменной %i удаляются обрамляющие кавычки (");
%%~fi - переменная %i расширяется до полного имени файла;
%%~di - из переменной %i выделяется только имя диска;
%%~pi - из переменной %i выделяется только путь к файлу;
%%~ni - из переменной %i выделяется только имя файла;
%%~xi - из переменной %i выделяется расширение имени файла;
%%~si - полученный путь содержит только короткие имена;
%%~ai - переменная %i принимает значение атрибутов файла;
%%~ti - переменная %i принимает значение даты /времени файла;

%%~zi - переменная %i принимает значение размера файла.
Возможно объединение нескольких операторов:
%%~dpi - переменная %i заменяется только на имя диска и путь;
%%~nxi - переменная %i заменяется только на имя файла и его расширение;
%%~fsIi - переменная %i заменяется только на полный путь с краткими именами;
%%~ftzai - переменная %I заменяется на строку, выдаваемую командой DIR.

Значение переменной **%%pi** внутри цикла команды FOR /R будет последовательно принимать значения путей папок, начиная с заданного набора **[диск:]путь**.

Так же, как и в предыдущем примере, желательно обнулить файл с результатами возможного предыдущего запуска данного командного файла:

```
REM Обнулить / создать файл для хранения списка каталогов C:\dirlist.txt
copy nul C:\dirlist.txt
REM Занесем первой строкой в пустой файл что-то вроде заголовка списка
Echo *** Список папок на диске C: *** >> C:\dirlist.txt
REM Сделать текущим каталогом корневой каталог диска C:
cd c:\
REM Выполнить для корневого каталога и всех вложенных каталогов, команду ECHO с
выдачей значения переменной %%~pi
for /R %%i in (C) DO (
ECHO Папка "%%~pi" >> C:\dirlist.txt
)
```

В результате выполнения этого командного файла в корне диска C: будет создан файл dirlist.txt, содержащий список каталогов диска.

Если в цикле команды **FOR /R** используются подстановочные значения переменной **%%I**, то в качестве набора **(in)** не стоит использовать символ точки.

Задача - найти на диске файлы с расширением .log и скопировать их в каталог на другом логическом диске - D:\MUSOR. Желательно проверить наличие каталога D:\MUSOR и при необходимости, создать его командой **md**, а также удалить из него все файлы, если они существуют, командой **del** . Затем выполнить переход в корневой каталог диска C: и выполнить в цикле команды FOR поиск файлов по маске ***.log** во всех подкаталогах.

```
REM подготовить каталог D:\MUSOR
if not exist D:\MUSOR md D:\MUSOR
REM удалить без подтверждения ( /Q) все файлы из каталога
del /Q D:\MUSOR\*.*
REM перейти в корень диска C:
cd c:\
REM Выполнить проверку наличия файлов с расширением *.log и скопировать их в
REM D:\MUSOR
for /R %%i in (c) DO (
if exist "%%~dpi*.log" copy "%%~dpi*.log" "D:\MUSOR\*.*"
)
```

Практика использования **FOR /R** показала, что не стоит использовать в качестве набора для обработки символ "точка" (конструкция **in (.)**), поскольку при использовании подстановочных значений, можно получить возврат из текущего каталога на уровень выше. В данном примере в качестве набора in используется любой не служебный символ. Команду копирования (copy) можно заменить на команду перемещения файлов (MOVE), что приведет к удалению файлов источников после копирования в каталог D:\MUSOR. Пример с копированием файлов с расширением .log рассмотренный выше имеет некоторые существенные недостатки - не обрабатываются скрытые файлы и папки, и в конечном каталоге, куда копируются файлы (D:\ MUSOR) не создаются подкаталоги с теми же именами, которые принадлежат путям исходных копируемых файлов. Для устранения этих недостатком можно использовать немного другой скрипт :

@echo off
REM подготовить каталог D:\MUSOR - удалить его и его подкаталоги командой RD
RD /S /Q D:\MUSOR
REM Создадим каталог заново
MD D:\MUSOR
REM Задаем начальную папку для обработки в команде FOR - C:\
for /R C:\ %%i in (C) DO (
xcopy "%%~dpi*.log" "D:\MUSOR%%~pi*.*" /H /R /Q /Y
)
Для копирования используется команда **xcopy** с ключами:
/H - копировать скрытые файлы.
/R - разрешение на замену файлов с атрибутом "Только чтение"
/Q - не отображать имена копируемых файлов
/Y - разрешать перезаписывать существующие файлы.
Подсказку по использованию команды **XCOPY** можно получить при вводе:
help xcopy
xcopy /?

При обработке строки **xcopy "%%~dpi*.log" "D:\MUSOR%%~pi*.*" /H /R /Q /Y** в цикле FOR, в качестве источника копирования будет выбираться **C:\текущий путь*.log** а в качестве приемника - **D:\MUSOR\текущий путь\имя копируемого файла**. Похожий подход можно использовать для обнаружения и копирования исполняемых файлов (*.exe) из каталога временных файлов, задаваемого переменной TEMP. Бывает полезно для поиска вредоносных программ.

REM @echo off
REM подготовить каталог D:\MUSOR - удалить командой RD
RD /S /Q D:\MUSOR
REM Создадим каталог заново
MD D:\MUSOR
REM Задаем начальную папку для обработки (%TEMP%) и выполняем FOR
for /R "%TEMP%" %%i in (C) DO (
xcopy "%%~dpi*.exe" "D:\MUSOR%%~pi*.*" /H /R /Q
)

При работе с содержимым каталогов удобно использовать команды запоминания текущего каталога и перехода в новый **PUSHD** и команды восстановления ранее запомненного текущего каталога **POPD:**

PUSHD "%TEMP%"
Echo Работаем в каталоге временных файлов
REM новый каталог стал текущим и можно использовать относительные пути
REM Выдать список exe-файлов текущего каталога (%TEMP%) командой DIR
DIR *.exe
REM Восстановить путь, запомненный командой PUSHD
POPD
)
Echo Вернулись в исходный каталог

15. Работа с графическими приложениями Windows

Допустим, вам нужно из одного и того же командного файла запустить notepad.exe и cmd.exe. Если просто вставить строки

notepad.exe

cmd.exe,

то после запуска notepad.exe выполнение командного файла приостановится и пока не будет завершен notepad, cmd.exe не запустится. Самый простой способ обойти эту проблему -

использовать стандартную команду Windows **start**. Полную справку по использованию можно получить по:

start /?

Создайте командный файл следующего содержания:

```
start /MAX notepad.exe
start "This is CMD.EXE" /MIN cmd.exe
net send %COMPUTERNAME% NOTEPAD and CMD running.
```

После выполнения этого командного файла вы увидите стартовавшие, в развернутом окне (ключ /MAX) блокнот, в свернутом окне (ключ /MIN) командный процессор CMD.EXE и окно с сообщением net.exe. Стандартный заголовок окна cmd.exe заменен на текст "This is CMD.EXE". Обратите внимание на то что заголовок окна можно опускать, но особенность обработки входных параметров командой start может привести к неожиданным результатам при попытке запуска программы, имя или путь которой содержит пробел(ы). Например при попытке выполнить следующую команду:

start "C:\Program Files\FAR\FAR.EXE"

Из-за наличия пробела в пути к исполняемому файлу, строка для запуска FAR.EXE должна быть заключена в двойные кавычки, однако формат входных параметров для start предполагает наличие заголовка окна, также заключаемого в двойные кавычки, в результате чего "C:\Program Files\FAR\FAR.EXE" интерпретируется не как исполняемая программа, а как заголовок окна. Для того, чтобы подобного не случилось нужно использовать любой, пусть даже пустой, заголовок:

start "" "C:\Program Files\FAR\FAR.EXE"

Если вам все же потребуется расширенное управление окнами приложений, придется воспользоваться сторонним программным обеспечением, например, **CMDOW.** Из-за специфического поведения эта утилита большинством антивирусов определяется как вирус, поэтому для нормальной работы нужно занести ее в исключения антивируса. **Cmdow.exe** - крошечная утилита, работающая в Windows NT4/2000/XP/2003 без установки. Позволяет получить список окон, перемещать, изменять размеры, переименовывать, сворачивать/разворачивать, активировать/деактивировать, закрывать, скрывать окна приложений и многое другое. Справку можно получить по команде: **cmdow /?**

Используется около 30 ключей. Некоторые примеры:

- **Получение информации об окнах:**

cmdow.exe или **cmdow.exe > wins.txt** - выдать информацию обо всех окнах на экран или в файл wins.txt

cmdow /T - выдать информацию об окнах, отображаемых на панели задач рабочего стола. Информация содержит колонки:

Handle - дескриптор окна - шестнадцатеричное число, связанное с данным окном.

Lev - уровень окна. Приложение может быть многооконным с несколькими уровнями окон.

Pid - идентификатор процесса, породившего окно.

-Window status - состояние окна (видимое - Vis, скрытое - Hid, активное - Act, свернутое - Min и т.п.

Image - программа вызвавшая окно.

Caption - название окна

Манипулировать окнами можно используя название окна, или его дескриптор. Если название окна содержит пробелы, то оно заключается в двойные кавычки. Если имеются русские буквы, то должна использоваться DOS-кодировка. Символ **@** используется для указания текущего окна. Иногда проще использовать дескриптор окна, а не его название. Полезным может быть и использование команды поиска по строке find.exe, выполняемой в цепочке с cmdow:

cmdow.exe | find.exe /I "hid" > wins.txt - в файл wins.txt попадут только строки содержащие шаблон "hid" и мы получим список скрытых окон. **cmdow.exe | find.exe /I "MyIE" > wins.txt** - список окон приложения MyIE

- **Манипулирование окнами**

Если вы хотите, чтобы ваш командный файл выполнялся скрытно, добавьте в него строку: **cmdow @ /HID** - скрыть текущее окно.

Ниже командный файл с комментариями, демонстрирующий возможности работы cmdow:

```
@ECHO OFF
REM Свернуть все окна - /MA
cmdow /MA
REM запустить cmd.exe с заголовком окна MyCMD
start "MyCMD" cmd.exe
REM ждать 5 секунд
call :wait5s
REM
:M1
REM Скрыть окно MyCND
cmdow MyCMD /hid
call :wait5s
REM Сделать видимым
cmdow MyCMD /vis
call :wait5s
REM Переместить в верхний левый угол экрана и развернуть окно
cmdow MyCMD /MOV 0 0
cmdow Mycmd /max
call :wait5s
REM Изменить размер на 320 x 240 и переместить вправо на 320 точек
cmdow MyCMD /MOV 320 0 /SIZ 320 240
call :wait5s
REM Переместить окно в точку с координатами 320 x 240 и изменить размер на 350x50
cmdow MYCMD /MOV 320 240 /SIZ 350 50
call :wait5s
REM Восстановить окно
cmdow MYCMD /RES
call :wait5s
REM Восстановить и сделать активным окно этого командного файла
cmdow @ /RES /ACT
ECHO Для завершения нажмите CTRL-C (CTRL-Break)
call :wait5s
call :wait5s
REM Зацикливание - переход к метке :M1
GOTO M1
REM Подпрограмма задержки на 5секунд
:wait5s
@ping -n 5 localhost > nul
```

Пример командного файла, закрывающего окна Проводника Интернет (IEXPLORE.EXE):

```
@echo off
:M1
for /f "tokens=1-2,8" %%a in ('cmdow') do (
if /i "%%c"=="IEXPLORE" if "%%b"=="1" cmdow %%a /END > nul
```

```
)
goto M1
```

Работает это следующим образом. Из выходных данных CMDOW берется первое, второе и 8-е поля. Первое - дескриптор окна (Handle), второе - уровень (Lev), третье - имя программы (Image). В цикле выполняется cmdow и если в ее выводе имеется строка, где имя программы IEXPLORE и уровень окна 1 выполняется **cmdow <дескриптор> /END**. Пока этот командный файл выполняется, запустить "Обозреватель интернета" не получится. А если в начало командного файла добавить "cmdow @ /hid" - то будет скрыто и его окно.

16. Перекодировка текстовых файлов

В рассматриваемом примере нужно преобразовать исходный текстовый файл в DOS-кодировке в новый текстовый файл в Windows-кодировке. В качестве механизма перекодировки используется смена кодовой страницы командой **CHCP** и построчная выдача содержимого исходного файла командой **ECHO** с перенаправлением вывода в новый файл. Для DOS-кодировки используется кодовая страница 866, для Windows-кодировки - 1251. В примере исходный файл называется 866.txt, а файл с перекодированными данными - 1251.txt:

```
@echo off
  chcp 866 >nul
  for /f "tokens=*" %%i in (866.txt) do call:to1251 "%%i"
  exit
:to1251
  chcp 1251 >nul
  echo %~1 >>1251.txt
  chcp 866 >nul
  exit /b
```

Аналогичный подход можно использовать и для преобразования текста из Windows - кодировки (кодовая страница 1251) в DOS-кодировку (кодовая страница 866). Естественно, такая перекодировка не может учитывать пустые строки и форматирование текста с помощью спецсимволов, поскольку команда ECHO не позволяет работать с такими форматами данных.

Своеобразным современным стандартом программы для перекодировки файлов считается, портированная из Unix утилита **iconv** (в составе библиотеки libiconv):

iconv [-c] [-s] [-f encoding] [-t encoding] [inputfile ...]

Входная кодировка задаётся ключом **-f**, а выходная - ключом **-t** . Если ключи не заданы, используется кодировка для языка системы по умолчанию. Все входные файлы читаются по очереди, если не задан параметр входного файла, то используется стандартный ввод, а конвертируемый текст выводится на стандартный вывод. Когда задана опция **-c**, символы, которые не могут быть преобразованы просто выбрасываются. В противном случае при появлении подобной ошибки программа аварийно завершается.

Когда задана опция **-s**, сообщения об ошибках не выводятся. Ключ **-l** позволяет получить список доступных кодировок. Утилита позволяет перекодировать текст, практически, из любой кодировки в любую.

17. Часто встречающиеся ошибки при написании командных файлов

- **Командный файл вручную выполняется успешно, но запущенный с помощью планировщика не работает.**

Обычно, это вызвано тем, что вы не учитываете тот факт, что на момент выполнения вашего командного файла переменные среды могут быть совсем другими, чем на момент его написания и запуска из командной строки. Например, в командном файле используется запуск приложения myprog.exe, находящегося в каталоге SCRIPTS на диске D: . Если в командном файле используется имя модуля без полного пути MYPROG.EXE и если каталог D:\SCRIPTS не прописан в путях поиска (переменная PATH) то модуль MYPROG.EXE может быть найден и

выполнен только если текущим каталогом является D:\SCRIPTS. Но если вы укажете полный путь к myprog.exe:

D:\SCRIPTS\myprog.exe,

то программа будет найдена и выполнена в любом случае. Кроме того, нередко программа, указанная в командном файле использует для поиска своих компонент (dll, ini и т.п.) собственный каталог. Но на момент ее выполнения текущим каталогом может быть любой (чаще всего - системный каталог Windows). Естественно, компоненты не находятся и программа не выполняется. Для устранения проблемы добавьте в командный файл команды, обеспечивающие переход в нужный каталог. Например, программа myprog.exe должна выполняться в каталоге D:\SCRIPTS:

Rem Сменим текущий диск

D:

Rem перейдем в каталог SCRIPTS
CD D:\SCRIPTS
myprog.exe

- **Неправильно отображаются русские имена файлов, служб и т.п.**

Причина в том, что при создании командных файлов вы использовали текстовый редактор, в котором русские символы представлены не в DOS-кодировке. Если в приведенном выше примере перезапуска службы "DNS-клиент" вы используете неверную кодировку, то русская часть имени службы не будет опознана из-за неверной кодировки и будет выдано сообщение, что указанная служба не установлена. Чтобы избежать проблем с русскими символами в командных файлах, используйте редактор с поддержкой DOS-кодировки, например, встроенный редактор файлового менеджера FAR. Переключение между кодировками в редакторе осуществляется нажатием **F8** . С помощью FAR можно легко осуществлять перекодировку, скопировав (вырезав) текст в буфер обмена, затем нажав F8 и вставив текст из буфера.

- **Командный файл выполняется на одном компьютере успешно, но на другом - не работает.**

Обычно это вызвано применением в командных файлах абсолютных значений вместо переменных среды окружения. Вместо C:\WINDOWS правильнее использовать %SYSTEMROOT%, потому, что на другом компьютере система может быть установлена в другой каталог или на другой диск. Старайтесь вместо имени командного файла использовать переменную %0 и ее подстановочные варианты (%~d0 - диск с которого запущен сценарий, %~dp0 - полный путь и т.д.). Строки с переменными, принимающими значения имен файлов и каталогов лучше заключать в кавычки. Командная строка

DIR %ProgramFiles%

не выдаст вам содержимого каталога C:\Program Files , поскольку из-за наличия пробела будет интерпретирована как DIR C:\Program. Командная строка DIR "%ProgramFiles%" выполнится верно. Старайтесь использовать комманды Setlocal и Endlocal, чтобы не оставлять мусор из переменных, созданных или модифицированных командным файлом **.<br.< span="" style="box-sizing: border-box;"></br.<>**

18. Использование командных файлов в сценариях регистрации пользователей

Командные файлы удобно использовать для выполнения каких-либо действий при регистрации пользователя в домене. Делается это с помощью вкладки **Profile** свойств пользователя домена.

<

Сами командные файлы должны находиться в сетевой папке **Netlogon** (WINDOWS\SYSVOL\ DOMAIN\SCRIPTS) контроллера домена.

1. **Порядок выполнения работы**

Составить командный файл для загрузки системы в минимальной конфигурации:

1. Включить команду для того, чтобы обрабатываемые командным процессором строки не выдавались на экран.
2. Включить команду для просмотра и редактирования командного файла, содержащего символы русского алфавита с использованием редактор с стандартного приложения "Блокнот" (notepad.exe).
3. Задать цвет фона и цвет символа.
4. Вывести справку в файл с именем help.txt.
5. Ввести команду задания и модификикации пути поиска исполняемых программ в каталоге **C:\Windows**, и, если результат неуспешен, в **C:\windows\system32;** указать количество процессоров; архитектуру процессора; идентификатор процессора; уровень (номер модели) процессора; версию процессора; формат приглашения командной строки; букву системного диска; каталог ОС Windows.
6. Реализовать просмотр действующего значения какой-либо переменной.

2. **Контрольные вопросы**

15. Дать понятие командного файла.
16. Как запустить командный процессор в интерактивном режиме?
17. Какая кодовая страница используется при работе в среде Windows?
18. Какая команда выводит полную справку?
19. Что такое переменные окружения?
20. Как передавать параметры командной строки и использовать их значения в операциях внутри самого командного файла?

3. **Отчет по выполненной работе**

Отчет должен содержать пример составленного командного файла.

Практическая работа №7

Работа с пакетными файлами. Конфигурирование файлов autoexec.bat и config.sys.

Цель работы: изучить команды работы с файлами в операционной системе MS-DOS

Результат обучения. После обучения студент должен:

- знать основные команды настройки операционной системы MS-DOS;
- уметь оптимизировать работу операционной системы MS-DOS.

План занятия:

1. Изучение теоретических вопросов темы 20 минут.
2. Выполнение практического задания 50 минут.
3. Выполнение отчета 20 минут.

1. Командные файлы

Для упрощения ввода длинной последовательности команд используются командные файлы. ***Командный файл*** - это текстовый файл (в коде ASCII), состоящий из группы команд MS-DOS. Командный файл всегда записывается на диск с расширением ".BAT". Выполнение командного файла можно прервать в любой момент, нажав на клавиши Ctrl-Break.

Для удобства пользователя в системе предусмотрен специальный командный файл **AUTOEXEC.BAT**. Если он находится в корневом директории рабочего диска, то при загрузке

MS-DOS автоматически выполняются его команды. Назначение файла - экономия времени (он состоит из обычно вводимых при загрузке команд).

Командный файл может быть создан с помощью любого текстового редактора. В командном файле могут использоваться следующие команды.

Команда **REM** служит исключительно для вывода сообщений на экран дисплея в процессе работы командного файла. Она вводится в командный файл вместе с требуемым сообщением. Длина сообщения не может превышать 123 символов. Пример:

REM Начало файла

Команда **ECHO** позволяет управлять выводом выполняемых команд в командных файлах. Команда ECHO вводится в командный файл в следующем виде. Набирается имя команды - "echo" и режим ее работы - on или off. При введении ECHO ON команды отображаются в обычном (описанном выше) режиме. ECHO OFF подавляет выдачу команд на экран, включая и команды rem. Однако на экране будут появляться все сообщения, генерируемые системой в процессе работы командного файла.

При отсутствии ECHO в командном файле по умолчанию работает режим "ON". Если произошло прерывание выполнения командного файла (аварийное или нормальное), то команда ECHO автоматически переходит в режим ON. При введении имени команды ("ECHO") на экране отображается режим ее работы в текущий момент времени.

Команда **PATH** - указывает, в каких каталога MS-DOS должна искать выполняемые файлы. По умолчанию маршрут поиска ограничивается только текущим каталогом. Общий вид команды:

PATH [[диск:]каталоги[;...]]

Максимальная длина команды PATH ограничена 127 символами. Если в маршруте PATH задается несколько каталогов, то они разделяются точкой с запятой, например:

path c:\user1\progs; c:\sys\suppl; d:\utils\dos

Команда **SET** выводит на экран, устанавливает или отменяет переменные операционной среды MS-DOS, которые используются для управления поведением некоторых командных файлов и программ, а также управляют работой MS-DOS. Вид команды:

SET [переменная=[строка]]

Параметр "переменная" задает устанавливаемую или модифицируемую переменную среды, "строка" задает связываемую с конкретной переменной строку. Например, для размещения временных файлов в определенном каталоге необходимо указать команду:

SET TMP=путь к каталогу

2. Файл настройки CONFIG.SYS

Файл CONFIG.SYS представляет собой текстовый файл, содержащий специальные команды. Эти команды настраивают конфигурацию компьютера таким образом, что его аппаратные компоненты могут использовать прикладные программы и сама операционная система MS-DOS. При запуске MS-DOS операционная система выполняет команды в файле CONFIG.SYS. Обычно это файл находится в корневом каталоге диска C.

В файле CONFIG.SYS можно использовать запросы на выполнение каждой команды. Делается это с помощью символа «**?**». Если вы хотите выводить запрос, укажите этот символ непосредственно после соответствующей команды (без пробелов), но перед знаком равенства (=). Например, DOS?=HIGH.

В файле CONFIG.SYS могут использоваться следующие команды.

Команда **BUFFERS** выделяет память для указанного числа дисковых буферов. Вид команды:

BUFFERS=n[,m]

Параметр n задает число дисковых буферов (от 1 до 99), m задает число буферов во вспомогательном буферном кэше (от 0 до 8). Например, чтобы задать 25 дисковых буферов, задайте в CONFIG.SYS команду buffers=25.

Если для m или n задано недопустимое значение, BUFFERS использует установку по умолчанию. По умолчанию число буферов во вспомогательном кэше (m) равно 0.

Использование кэш-буфера ускоряет определенные операции с диском при работе на компьютере. MS-DOS использует зарезервированную для каждого дискового буфера память для данных при операциях чтения и записи. Для получения лучшей производительности при работе с такими программами, как текстовые редакторы, задайте для n значение от 20 до 30. Если вы собираетесь создавать много подкаталогов, то лучше увеличить число буферов до 40 - 50.

Команда **DEVICE** - загружает драйверы устройств в память компьютера. Вид команды:

DEVICE=[диск:][маршрут]имя_файла [параметры]

С "мышью", дисководом CDROM, сканером или другим аналогичным продуктов производитель обычно поставляет соответствующее программное обеспечение - драйвер устройства. Чтобы установить драйвер, задайте в командной строке DEVICE его расположение (каталог) и имя.

Команда **DEVICEHIGH** - загружает драйверы устройств в старшую память (> 640K). Это позволяет освободить обычную память для других программ. Если старшая память недоступна, то команда DEVICEHIGH работает аналогично команде DEVICE. Вид команды:

DEVICEHIGH [диск:][маршрут]имя_файла [параметры]

Чтобы использовать команду DEVICEHIGH, в файл CONFIG.SYS нужно включить команду DOS=UMB. Если вы не зададите эту команду, то все драйверы устройств будут загружаться в обычную память, как при задании команды DEVICE.

Для загрузки драйвера устройства в старшую память ваш компьютер должен иметь расширенную память. Сначала вы должны задать команду DEVICE для загрузки драйвера HIMEM.SYS, а затем указать ее снова для программы работы с UMB. Эти команды должны следовать в CONFIG.SYS до команды DEVICEHIGH. Пример:

```
device=c:\dos\himem.sys
dos=umb
```

Команда **FILES** - задает число файлов, одновременно доступных MS-DOS. Вид команды:

FILES=x

Параметр x задает число одновременно доступных файлов. Допускаются значения от 8 до 255 (по умолчанию - 8). Некоторые программы требуют большего значения, чем установленное по умолчанию, поэтому лучше установить значение 30-50. Установленное в FILES значение может быть не единственным определяющим фактором числа файлов, которые может одновременно открыть программа. Это число может также ограничиваться построением программы.

Команда **DOS** определяет, что MS-DOS должна поддерживать связь со старшей памятью, частично загружаться в старшую память (HMA) или и то, и другое. Вид команды:

DOS=HIGH, [LOW, UMB, NOUMB]

Параметр UMB определяет, что MS-DOS должна поддерживать блоки старшей памяти, созданные программой поддержки UMB, такой как EMM386.EXE. Параметр NOUMB задает,

что MS-DOS не должна обслуживать блоки UMB. По умолчанию используется параметр NOUMB.

Параметр HIGH определяет, должна ли MS-DOS пытаться загружать свою часть в HMA (HIGH) или загружать все свои компоненты в обычную память (LOW). По умолчанию устанавливается LOW.

Перед заданием DOS=UMB или DOS=HIGH вы должны установить драйвер или другой администратор расширенной памяти.

Файлы CONFIG.SYS и AUTOEXEC.BAT. DOS

Основную роль в установлении конфигурации DOS играют файлы CONFIG.SYS и AUTOEXEC.BAT. DOS, при начальной загрузке считывает из корневого каталога загрузочного диска файлы CONFIG.SYS и AUTOEXEC.BAT и выполняет содержащиеся там команды. Файл CONFIG.SYS является текстовым файлом, в котором содержатся специальные команды для настройки конфигурации DOS: подключения различных драйверов, определения размеров системных таблиц DOS и т.д. Заданные в файле CONFIG.SYS команды выполняются в процессе начальной загрузки DOS.

После завершения выполнения файла CONFIG.SYS автоматически выполняется командный файл AUTOEXEC.BAT, если он имеется в корневом каталоге загрузочного диска. Как правило, в файл AUTOEXEC.BAT записывают команды для запуска резидентных программ и других программ, которые целесообразно запускать при каждой загрузке DOS, а также команды для установки переменных окружения DOS (команда Set), задания списка каталогов, в которых производится поиск запускаемых программ (команда Path), и установки формата приглашения DOS (команда Prompt).

Файл HIMEM.SYS и emm386.sys

HIMEM - это администратор дополнительной памяти, программа, координирующая использование дополнительной памяти вашего компьютера, включая старшую память (HMA), благодаря чему никакие прикладные программы и драйверы устройств не используют одновременно одну и ту же память.

Установить HIMEM вы можете с помощью команды DEVICE для HIMEM.SYS в файле CONFIG.SYS. Команды для HIMEM.SYS должны следовать перед другими командами, запускающими прикладные программы или драйверы устройств, использующие дополнительную память (например, EMM386.EXE)

emm386.sys - обеспечивает доступ к старшим областям памяти и использует дополнительную память для моделирования расширенной памяти. Этот драйвер устройства должен загружаться командой DEVICE в файле CONFIG.SYS и может использоваться только на компьютерах с процессором 80386 или старше. EMM386 позволяет также загружать драйверы устройств и программы в блоки старшей памяти (UMB)

3. Практическое задание

Задание №1. Создание командного файла загрузки autoexec.bat

1. Перезагрузил компьютер в режиме MS-DOS.
2. Перешел в корневой каталог диска **C**.
3. На диске **C** создал каталог SYSTEM.
4. Внутри этого каталога создал текстовый файл **autoexec.bat**.
5. Внесите в этот файл в текстовом виде следующие команды:

- Указал комментарий: «Пример командного файла».
- Отключил вывод на экран сообщений при исполнении файла..
- Установил в качестве пути поиска выполняемых файлов следующие каталоги: WINDOWS, WINDOWS\SYSTEM, MOUSE, TOOLS, находящиеся на диске C и каталоги MASTER, TEMP\ COMMAND, находящиеся на диске D.

- Указал расположение временных файлов в папке TEMP на диске D.
- Указал запуск указателя мышки (файл MOUSE.COM в папке MOUSE на диске C).
- Указал запуск русификатора (файл KEYRUS.COM в папке KEYRUS на диске C).
- Указал, что при загрузке должна автоматически запустится программа Norton Commander (файл NC.EXE в папке NC на диске C).
- Установил комментарий для строки запуска русификатора.

6. Сохранил файл и выйдите из режима редактирования.

Содержание файла autoexec.bat

```
Rem Пример командного файла
Echo off
Path c:\windows;
C:\windows\system;
C:\mouse;
C:\tools;
D:\master;
Set tmp= d:\temp
Set mouse.com= c:\mouse
Set keyRus.com= c:\keyRus
Set nc.exe= c:\nc
Rem Запуск русификатора
```

Задание №2. Создание файла конфигурации config.sys.

1. Внутри каталога **SYSTEM** создал файл **config.sys**.
2. Внес в него следующие команды:

- Указал, чтобы модули ОС загружались в верхнюю память HMA.
- Указал, чтобы модули ОС загружались в область памяти UMB.
- Установил максимальное число одновременно открытых файлов – 40.
- Установил максимальное число промежуточных буферов – 20.
- Указал загрузку драйвера HIMEM.SYS из папки MSDOS на диске C.
- Указал загрузку драйвера EMM386.SYS из папки MSDOS на диске C.
- Указал загрузку драйвера мышки (файл MOUSE.SYS в папке MOUSE на диске C).
- Установил комментарий для строки загрузки драйвера мышки.

3. Сохраните файл и выйдите из режима редактирования.

Содержание файла config.sys

```
Dos=high
Dos=umb
Files=40
Buffers=20
Device=c:\msdos\himeme.sis
Device=c:\msdos\emm386.sys
Device=c:\mouse\mouse.sys
Rem Загрузка драйвера мыши.
```

Задание №3. Создание произвольного командного файла.

1. Внутри каталога **SYSTEM** создал файл **pusk.bat**.
2. Внес в него команды, чтобы автоматически было выполнено следующее задание:

- Создал каталог MASTER.
- Внутри каталога MASTER создал три каталога VOPROS, OTVET, OTHERS.
- Скопировал файл PRINTERS.TXT из папки WINDOWS в папку MASTER.
- Скопировал файл MOUSE.TXT из папки WINDOWS в папку MASTER.
- Объединил файлы PRINTERS.TXT и MOUSE.TXT в единый файл ALL.TXT.
- В папке MASTER файл PRINTERS.TXT переименовал в файл MACHINES.TXT.
- В папке MASTER файл MOUSE.TXT переименовал в файл KAT.TXT.
- Удалил каталог OTHERS.
- Скопировал все файлы из каталога C:\WINDOWS\COMMAND в каталог SYSTEM\ MASTER\VOPROS
- В папке MASTER у всех файлов изменил расширения с TXT на XTX.

3. Сохранил файл и выйдите из режима редактирования.
4. Показал результат работы преподавателю.

Содержание файла pusk.bat

```
Cd C:\system
Md C:\system\master
Md otvet
Md vopros
Md others
Copy C:\windows\printers.txt C:\system\master
Copy C:\windows\mose.txt C:\system\master
Cd C:\system\master
Copy printers.txt+mose.txt All.txt
Ren Printers.txt Machinex.txt
Ren mouse.txt kat.txt
Rm dir others
Copy C:\windows\command\*.* C:\system\master\vopros
Ren C:\system\master\*.txt *.xtx
```

Практичесая работа № 8
Управление процессами

Цель работы: Изучение процессов в операционной системе Linux.

Теоретическая часть

Лабораторная работа посвящена процессам операционной системы Linux. Поскольку администрирование операционной системы в конечном счете сводится к управлению процессами, каждый раз, когда программа запускается на выполнение, начинается то, что в

литературе именуется как *процесс*. Или другими словами – процессом называется выполняемая в данный момент программа или ее потомки. Каждый процесс запускается от имени какого-то пользователя. Процессы, которые стартовали при загрузке, обычно выполняются от имени пользователей root или nobody.

Каждый пользователь может управлять поведением процессов, им запущенных. При этом пользователь root может управлять всеми процессами — как запущенными от его имени, так и процессами, порожденными другими пользователями операционной системы. Управление процессами осуществляется с помощью утилит, а также посредством некоторых команд командной оболочки (shell).

Каждый процесс в системе имеет уникальный номер — идентификационный номер процесса (Process Identification, PID). Этот номер используется ядром операционной системы, а также некоторыми утилитами для управления процессами.

Выполнение процесса на переднем плане и в фоновом режиме

Процессы могут выполняться на переднем плане (foreground) – режим по умолчанию и в фоновом режиме (background).

На переднем плане в каждый момент для текущего терминала может выполняться только один процесс. Однако пользователь может перейти в другой виртуальный терминал и запустить на выполнение еще один процесс, а на другом терминале еще один и т.д. Процесс переднего плана – это процесс, с которым взаимодействует пользователь, процесс получает информацию с клавиатуры (стандартный ввод) и посылает результаты на экран (стандартный вывод).

Фоновый процесс после своего запуска благодаря использованию специальной команды командной оболочки отключается от клавиатуры и экрана, т.е. не ожидает ввода данных со стандартного ввода и не выводит информацию на стандартный вывод, а командная оболочка не ожидает окончания запущенного процесса, что позволяет пользователю немедленно запустить еще один процесс.

Обычно фоновые процессы требуют очень большого времени для своего завершения и не требуют вмешательства пользователя во время существования процесса. К примеру, компиляция программ или архивирование большого объема информации — кандидаты номер один для перевода процесса в фоновый режим.

Процессы так же могут быть отложенными. Отложенный процесс — это процесс, который в данный момент не выполняется и временно остановлен. После того как пользователь остановил процесс, в дальнейшем может его продолжить как на переднем плане, так и в фоновом режиме. Возобновление приостановленного процесса не изменит его состояния — при возобновлении он начнется с того места, на котором был приостановлен.

Для выполнения программы в режиме переднего плана достаточно просто набрать имя программы в командной строке и запустить ее на выполнение. После этого пользователь может работать с программой.

Для запуска программы в качестве фонового процесса достаточно набрать в командной строке имя программы и в конце добавить знак амперсанта (&), отделенный пробелом от имени программы и ее параметров командной строки, если таковые имеются. Затем программа запускается на выполнение. В отличие от запуска программы в режиме переднего плана получается приблизительно следующее сообщение:

```
/home/student# yes > /dev/null &;
[1] 123
/home/student#
```

Оно состоит из двух чисел и приглашения командной строки. Таким образом, мы запустили программу выполняться в фоновом режиме и получили возможность запустить с той же самой консоли на выполнение еще какую-то программу.

Число [1] означает номер запущенного нами фонового процесса. С его помощью можно будет производить манипуляции с фоновым процессом. Значение 123 показывает идентификационный номер (PID) процесса. Отличия этих двух чисел достаточно существенные. Номер фонового процесса уникален только для пользователя, запускающего данный фоновый процесс. То есть если в системе три пользователя решили запустить фоновый процесс (первый для текущего сеанса) – в результате у каждого пользователя появится фоновый процесс с номером [1]. Напротив, идентификационный номер процесса (PID) уникален для всей операционной системы и однозначно идентифицирует в ней каждый процесс.

Номер фонового процесса хранится в переменных командной оболочки пользователя и позволяет не забивать голову цифрами типа 2693 или 1294, а использовать переменные вида %1, %2. Однако допускается пользоваться и идентификационным номером процесса.

Для проверки состояния фоновых процессов можно воспользоваться командой командной оболочки – jobs.

```
/home/student# jobs
[1]+  Running yes >/dev/null    &
/home/student#
```

Из вышеприведенного примера видно, что у пользователя в данный момент запущен один фоновый процесс, и он выполняется.

Остановка и возобновление процесса

Помимо прямого указания выполнять программу в фоновом режиме, существует еще один способ перевести процесс в фоновый режим. Для этого необходимо выполнить следующие действия:

- запустить процесс выполняться на переднем плане;
- остановить выполнение процесса;
- продолжить процесс в фоновом режиме.

Для выполнения программы введем ее имя в командной строке и запустим на выполнение. Для остановки выполнения программы необходимо нажать на клавиатуре следующую комбинацию клавиш — <Ctrl>+<Z>. После этого можно увидеть на экране следующее:

```
/home/student#  yes > /dev/null
ctrl+Z
[1]+  Stopped yes  >/dev/null
/home/student#
```

Получено приглашение командной строки. Для того чтобы перевести выполнение процесса в фоновый режим, необходимо выполнить следующую команду:

```
bg %1
```

Причем необязательно делать это сразу после остановки процесса, главное правильно указать номер остановленного процесса.

Для того чтобы вернуть процесс из в фонового режима выполнения на передний план, достаточно выполнить следующую команду:

```
fg %1
```

В том случае, если необходимо перевести программу в фоновый или, наоборот, на передний план выполнения сразу после остановки процесса, можно выполнить соответствующую программу без указания номера остановленного процесса.

Существует большая разница между фоновым и остановленным процессом. Остановленный процесс не выполняется и не потребляет ресурсы процесса, однако занимает оперативную память или пространство свопинга. В фоновом же режиме процесс продолжает выполняться.

Для приостановления фонового процесса необходимо переместить процесс на передний план, а затем остановить.

Завершение работы процесса

Существует несколько вариантов завершения работы процесса.

Вариант первый. Если процесс интерактивный, как правило, в документации или прямо на экране написано, как корректно завершить программу.

Вариант второй. В том случае, если не указано как можно завершить текущий процесс (не фоновый), то можно воспользоваться клавиатурной комбинацией <Ctrl>+<C>. Возможно использовать также комбинацию клавиш <Ctrl>+<Break>. А для остановки фонового процесса можно перевести его на передний план, а затем уже воспользоваться вышеприведенными клавиатурными комбинациями.

Вариант третий и самый действенный. В том случае, если не удалось прекратить выполнение процесса вышеприведенными способами – например, программа зависла или "слетел" терминал — для завершения процесса можно воспользоваться следующими командами: kill, killall.

Команда kill может получать в качестве аргумента как номер процесса, так и идентификационный номер (PID) процесса. Таким образом, команда:

/home/student# kill 123 эквивалентна команде:

/home/student# kill %1

Можно видеть, что не надо использовать "%", когда вы обращаетесь к работе по идентификационному номеру (PID) процесса.

С помощью команды killall можно прекратить выполнение нескольких процессов сразу, имеющих одно и то же имя. Например, команда killall mc прекратит работу всех программ mc, запущенных от имени данного пользователя.

Для того чтобы завершить работу процесса, пользователю надо быть его владельцем. Пользователь root может завершить работу любого процесса в операционной системе.

Программы, используемые для управления процессами

Существует достаточно большое количество утилит, используемых для управления тем или иным способом процессами, исполняемыми в операционной системе. Рассмотрим только основные утилиты. В табл. 1 приведен список основных программ, тем или иным образом предназначенных для управления процессами.

Таблица1. Программы управления процессами

Программа	Описание
at	Выполняет команды в определенное время
batch	Выполняет команды тогда, когда это позволяет загрузка системы
cron	Выполняет команды по заранее заданному расписанию
crontab	Позволяет работать с файлами crontab отдельных пользователей
kill	Прекращает выполнение процесса
nice	Изменяет приоритет процесса перед его запуском
nohup	Позволяет работать процессу после выхода пользователя из системы
ps	Выводит информацию о процессах
renice	Изменяет приоритет работающего процесса
w	Показывает, кто в настоящий момент работает в системе и с какими про граммами

nohup

Эта утилита позволяет организовать фоновый процесс, продолжающий свою работу даже тогда, когда пользователь отключился от терминала, в отличие от команды &, которая этого не позволяет. Для организации фонового процесса необходимо выполнить следующую команду:

nohup выполняемая_фоновая_команда &

Во вновь запущенном терминале процесс нельзя увидеть с помощью команды jobs, так как команда jobs выводит список процессов текущего терминала, поэтому после подключения к терминалу необходимо использовать команду ps с параметром –A.

ps

Программа ps предназначена для получения информации о существующих в операционной системе процессах. У этой команды есть множество различных опций, но остановимся на самых часто используемых. Для получения подробной информации смотрите man-страницу этой программы.

Простой запуск ps без параметров выдаст список программ, выполняемых на терминале. Обычно этот список очень мал:

PID	TTY	TIME	CMD
885	ttyl	00:00:00	login
893	ttyl	00:00:00	bash
955	ttyl	00:00:00	ps

Первый столбец – pid (идентификационный номер процесса). Как уже упоминалось, каждый выполняющийся процесс в системе получает уникальный идентификатор, с помощью которого производится управление процессом. Каждому вновь запускаемому на выполнение процессу присваивается следующий свободный PID. Когда процесс завершается, его номер освобождается. Когда достигнут максимальный PID, следующий свободный номер будет взят из наименьшего освобожденного.

Следующий столбец – tty – показывает, на каком терминале процесс выполняется. Запуск команды без параметров ps покажет процессы, выполняемые на текущем терминале.

Столбец time показывает, сколько процессорного времени выполняется процесс. Оно не является фактическим временем с момента запуска процесса, поскольку Linux – это многозадачная операционная система. Информация, указанная в столбце time, показывает время, реально потраченное процессором на выполнение процесса.

Столбец CMD(COMMAND) показывает имя программы. Отображается только имя, опции командной строки не выводятся.

Для получения расширенного списка процессов, выполняемых в системе, используется следующая команда:

ps –ax

Опции, заданные программе в этом примере, заставляют ее выводить не только имена программ, но и список опций, с которыми были запущены программы.

Появился новый столбец - STAT. В этом столбце отображается состояние (status) процесса. Полный список состояний вы можете прочитать в описании программы ps. Опишем ключевые состояния:

– буква R обозначает запущенный процесс, исполняющийся в данный момент времени;

– буква S обозначает спящий (sleeping) процесс — процесс ожидает какое-то событие, необходимое для его активизации;

– буква Z используется для обозначения "зомбированных" процессов (zombied) — это процессы, родительский процесс которых прекратил свое существование, оставив дочерние процессы рабочими.

Если обратить внимание на колонку TTY, то можно заметить, что многие процессы, расположенные в верхней части таблицы, в этой колонке содержат знак "?" вместо терминала. Так обозначаются процессы, запущенные с более не активного терминала. Как правило, это всякие системные сервисы.

Если вы хотите увидеть еще больше информации о выполняемых процессах, попробуйте выполнить команду:

ps -aux

Появились еще следующие столбцы:

- USER - показывает, от имени какого пользователя был запущен данный процесс;
- %CPU, %MEM - показывают, сколько данный процесс занимает соответственно процессорного времени и объем используемой оперативной памяти;
- TIME - время запуска программы.

top

Еще одна утилита, с помощью которой можно получать информацию о запущенных в операционной системе процессах. Для использования достаточно просто запустить команду top на выполнение. Эта утилита выводит на экран список процессов в системе, отсортированных в порядке убывания значений используемых ресурсов.

В начале сообщения, выведенного на экране, идет общесистемная информация — из нее можно узнать время запуска операционной системы, время работы операционной системы от момента последнего перезапуска системы, количество зарегистрированных в данный момент в операционной системе пользователей, а также минимальную, максимальную и среднюю загрузку операционной системы. Помимо этого, отображается общее количество процессов и их состояние, сколько процентов ресурсов системы используют пользовательские процессы и системные процессы, использование оперативной памяти и «свопа».

Далее идет таблица, во многом напоминающая вывод программы ps. Идентификационный номер процесса, имя пользователя — владельца процесса, приоритет процесса, размер процесса, его состояние, используемые процессом оперативная память и ресурс центрального процесса, время выполнения и, наконец, имя процесса.

Утилита top после запуска периодически обновляет информацию о состоянии процессов в операционной системе, что позволяет динамически получать информацию о загрузке системы.

kill

Программа kill предназначена для посылки соответствующих сигналов указанному процессу. Как правило, это бывает тогда, когда некоторые процессы начинают вести себя неадекватно. Наиболее часто программа применяется, чтобы прекратить выполнение процессов.

Для того чтобы прекратить работу процесса, необходимо знать PID процесса либо его имя. Например, чтобы "убить" процесс 123, достаточно выполнить следующую команду:

kill 123

Как обычно, чтобы прекратить работу процесса, пользователю необходимо быть его владельцем. Пользователь root может прекратить работу любого процесса в системе.

Иногда стандартное выполнение программы kill не справляется с поставленной задачей. Обычно это объясняется тем, что данный процесс завис либо выполняет операцию, которую с его точки зрения нельзя прервать немедленно. Для прерывания этого процесса можно воспользоваться следующей командой:

kill -9 123 .

Вообще-то программа kill предназначена для посылки процессам управляющих сигналов, одним из которых является сигнал sigterm (terminate, завершиться). Этот сигнал посылается процессу при выполнении программы kill по умолчанию. Процесс, получивший

данный сигнал, должен корректно завершить свою работу (закрыть используемые файлы, сбросить буферы ввода/вывода и т. п.). Ключ -9 указывает программе kill посылать процессу другой тип сигнала — sigkill. Это приводит к тому, что процесс не производит корректного завершения, а немедленно прекращает свою жизнедеятельность. Помимо этих сигналов, в вашем распоряжении целый набор различных сигналов. Полный список сигналов можно получить с помощью команды вызова помощи.

killall

Еще один вариант программы kill. Используется для того, чтобы завершить работу процессов, носящих одно и то же имя. К примеру, в системе запущено несколько программ mc. Для того чтобы одновременно завершить работу этих программ, достаточно всего лишь выполнить следующую команду:

killall mc

Конечно, этим не ограничивается использование данной команды. С ее помощью можно отсылать сигналы группе одноименных процессов. Для получения более подробной информации по этой команде обращайтесь к ее man-странице.

Изменение приоритета выполнения процессов

В операционной системе Linux у каждого процесса есть свой приоритет исполнения. Это очень удобно. Поскольку операционная система многозадачная – то для выполнения каждого процесса выделяется определенное количество времени. Для некоторых задач необходимо выделить побольше, для некоторых можно поменьше. Для этого и предназначен приоритет процесса. Управление приоритетом процесса осуществляется программами nice и renice.

nice

Программа nice позволяет запустить команду с предопределенным приоритетом выполнения, который задается в командной строке. При обычном запуске все задачи имеют один и тот же приоритет, и операционная система равномерно распределяет между ними процессорное время. Однако с помощью утилиты nice можно понизить приоритет какой-либо задачи, таким образом, предоставляя другим процессам больше процессорного времени. Повысить приоритет той или иной задачи имеет право только пользователь root. Синтаксис использования nice следующий:

nice -*number command*

Уровень приоритета процесса определяется параметром *number*, при этом большее его значение означает меньший приоритет процесса. Значение по умолчанию — 10, и *number* представляет собой число, на которое должен быть уменьшен приоритет.

К примеру, процесс top имеет приоритет, равный -5. Для того чтобы понизить приоритет выполнения процесса на десять, мы должны выполнить следующую команду:

nice 10 top

В результате процесс top имеет приоритет, равный 5.

Только пользователь root может поднять приоритет того или иного процесса, используя для этого *отрицательное* значение параметра *number*.

renice

Программа renice, в отличие от программы nice, позволяет изменить приоритет уже работающего процесса. Формат запуска программы следующий:

renice -*number PID*

В общем, программа renice работает точно так же, как и nice. Уровень приоритета процесса определяется параметром *number*, при этом большее его значение означает меньший приоритет процесса. Значение по умолчанию — 10, и *number* представляет собой число, на которое должен быть уменьшен приоритет процесса.

Только пользователь root может поднять приоритет того или иного процесса, используя для этого *отрицательное* значение параметра number.

Выполнение процессов в заданное время

Одна из основных задач автоматизации администрирования операционной системы — выполнение программ в заданное время или с заданной периодичностью. Для решения этих проблем существует несколько утилит, позволяющих запускать процессы в нужное время.

at

Для запуска одной или более команд в заранее определенное время используется команда at. В этой команде вы можете определить время и дату запуска той или иной команды. Команда at требует, по меньшей мере, двух параметров — время выполнения программы и запускаемую программу с ее параметрами запуска.

Приведенный ниже пример запустит команду на выполнение в 01:01. Для этого введите все, приведенное ниже, с терминала, завершая ввод каждой строки нажатием клавиши <Enter> и по окончании ввода всей команды — <Ctrl>+<D> для ее завершения.

```
at 1:01
ls
echo “Time is 1:01”
```

Помимо времени, в команде at может быть также определена и дата запуска программы на выполнение.

Пользователь root может без ограничения применять практически любые команды. Для обычных пользователей права доступа к команде at определяются файлами /etc/at.allow и /etc/at.deny. В файле /etc/at.allow содержится список тех, кому разрешено использовать команду at, а в файле /etc/at.deny находится список тех, кому ее выполнять запрещено.

batch

Команда batch в принципе аналогична команде at. Более того, batch представляет собой псевдоним команды at -b. Пользователь хочет запустить резервное копирование вечером. Однако в это время система очень занята, и выполнение резервирования системы практически парализует ее работу. Для этого и существует команда batch — ее использование позволяет операционной системе самой решить, когда наступает подходящий момент для запуска задачи в то время, когда система не сильно загружена.

Формат команды batch представляет собой просто список команд для выполнения, следующих в строках за командой; заканчивается список комбинацией клавиш <Ctrl>+<D>. Можно также поместить список команд в файл и перенаправить его на стандартный ввод команды batch.

cron

Cron — это программа, выполняющая задания по расписанию, но, в отличие от команды at, она позволяет выполнять задания неоднократно. Вы определяете времена и даты, когда должна запускаться та или иная программа. Времена и даты могут определяться в минутах, часах, днях месяца, месяцах года и днях недели.

Программа cron запускается один, раз при загрузке системы. При запуске cron проверяет очередь заданий at и задания пользователей в файлах crontab. Если для запуска не было найдено заданий — следующую проверку cron произведет через минуту.

Практическая часть

1. Запустите программу yes в фоновом режиме с подавлением потока вывода.
2. Запустите программу yes на переднем плане с подавлением потока вывода. Приостановите выполнение программы. Заново запустите программу yes с теми же параметрами, и завершите ее

выполнение.

3. Запустите программу yes на переднем плане без подавления потока вывода. Приостановите выполнение программы. Заново запустите программу yes с теми же параметрами, и завершите ее выполнение.
4. Проверьте состояния процессов, воспользовавшись командой jobs.
5. Переведите процесс, который у вас выполняется в фоновом режиме на передний план и остановите его.
6. Переведите любой ваш процесс с подавлением потока вывода в фоновый режим.
7. Проверьте состояния процессов, воспользовавшись командой jobs. Обратите внимание, что процесс стал выполняющемся (Running) в фоновом режиме.
8. Запустите процесс в фоновом режиме, таким образом, чтобы он продолжил свою работу даже после отключения от терминала.
9. Закройте окно и заново запустите консоль. Убедитесь, что процесс продолжил свою работу.
10. Получите информацию о запущенных в операционной системе процессах с помощью утилиты top.
11. Запустите еще три программы yes в фоновом режиме с подавлением потока вывода.
12. «Убейте» два процесса: для одного используйте его PID, а для другого его идентификатор конкретного задания.
13. Попробуйте послать сигнал 1 (SIGHUP) процессу, запущенному с помощью nohup и обычному процессу.
14. Запустите еще несколько программ yes в фоновом режиме с подавлением потока вывода.
15. Завершите их работу одновременно, используя команду killall.
16. Запустите программу yes в фоновом режиме с подавлением потока вывода. Используя утилиту nice, запустите программу yes с теми же параметрами и с приоритетом, большим на 5. Сравните абсолютные и относительные приоритеты у этих двух процессов.
17. Используя утилиту renice, измените приоритет у одного из потоков yes таким образом, чтобы у обоих потоков приоритеты были равны.
18. Сделайте так, чтобы в xx минут xx часов автоматически выполнилась утилита ls и вывелась строка текста «Lab rab 5. Zadanie 18». Учтите, что вывод будет осуществляться не на экран, а в файл: /var/spool/mail/student (xx минут xx часов – ближайшие несколько минут).
19. Сделайте так, чтобы в xx минут xx часов каждую пятницу автоматически выполнилась утилита ps и вывелась строка текста «Lab rab 5. Zadanie 19». Учтите, что вывод будет осуществляться не на экран, а в файл: /var/spool/mail/student (xx минут xx часов – ближайшие несколько минут).
20. Сделайте так, чтобы в xx минут xx часов каждый xx месяц автоматически выполнилась утилита ps и вывелась строка текста «Lab rab 5. Zadanie 20». Учтите, что вывод будет осуществляться не на экран, а в файл: /var/spool/mail/student (xx минут xx часов – ближайшие несколько минут, xx месяц – текущий месяц).

Контрольные вопросы.

1. Объясните, что произойдет, если запустить программу yes в фоновом режиме без подавления потока вывода.
2. Объясните разницу между действием сочетаний клавиш ^Z и ^C.
3. Опишите, что значит каждое поле вывода команды jobs.
4. Назовите главное отличие утилиты top от ps.
5. Почему процесс, запущенный с помощью nohup не «убивается» сигналом 1?

Практическая работа №9

Управление дисками. Оптимизация и дефрагментация.

Цель работы: научиться управлять разделами жесткого диска с помощью командной строки, обслуживать и оптимизировать жесткие диски

Теоретическая часть

Основы DISKPART

DiskPart — это инструмент для работы с дисками, разделами и томами в операционной системе Windows. При помощи DiskPart можно выполнить такие важные операции, как преобразование типов дисков, создание разделов и томов, конфигурирование RAIDмассивов. Помимо этого, DiskPart служит для настройки автоматического монтирования новых дисков в файловой системе, для назначения букв дискам и путей подключенным сетевым дискам. Однако DiskPart не предназначен для форматирования дисков. С этой целью применяется команда FORMAT.

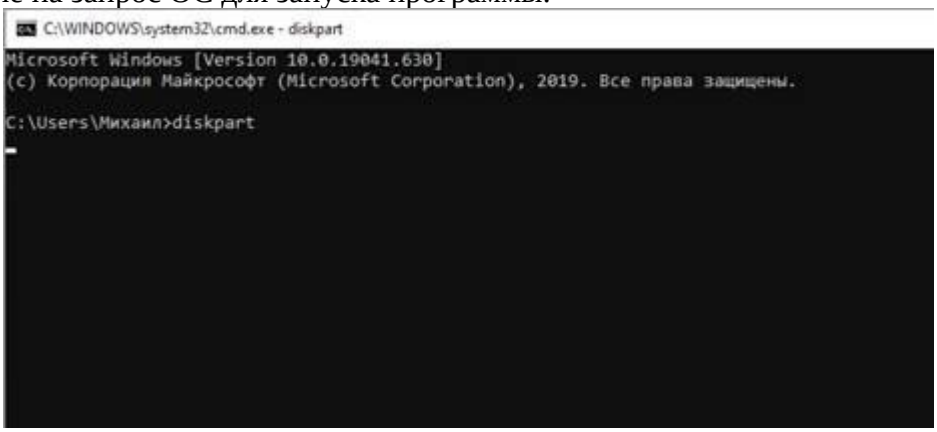
Запуск утилиты DISKPART

В отличие от других утилит командной строки, **DiskPart** — не простая утилита, запускаемая командной строкой с параметрами. Это скорее консольный интерпретатор команд со своей командной строкой и набором внутренних команд. Запускается DiskPart вводом команды «**diskpart»** в командной строке. С помощью этой утилиты возможно преобразование разделов жесткого диска, а также установка нового жесткого диска в систему и многое другое.

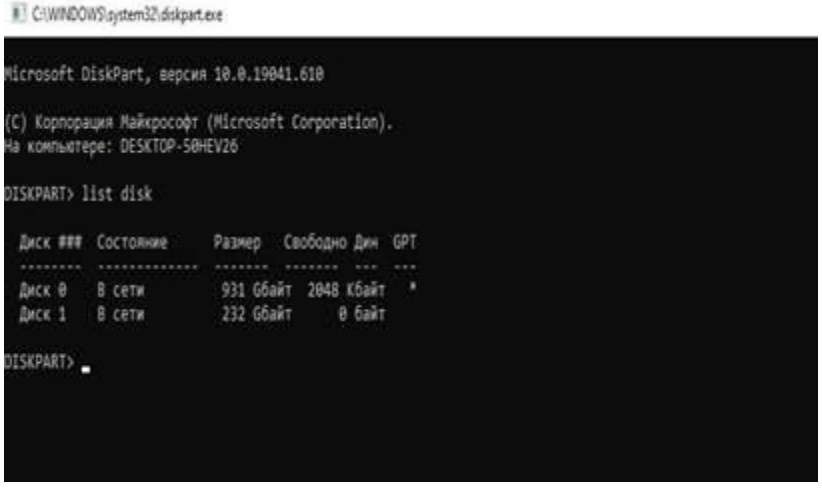
DiskPart работает с физическими жесткими дисками, установленными в компьютере. CD/DVD-приводы, съемные носители или подключаемые к USB-портам карты флэш-памяти не поддерживаются. Прежде чем использовать команды DiskPart, нужно перечислить, а затем выбрать диск, раздел или том, с которым вы хотите работать, для передачи ему фокуса. Когда диск, раздел или том находится в фокусе, любые команды DiskPart воздействуют именно на этот диск, раздел или том.

Ход практической работы

1. Для запуска утилиты запустите командную строку в ОС Windows и наберите команду «**diskpart**». Дайте разрешение на запрос ОС для запуска программы.



2. Прежде чем создавать новые разделы нам надо убедиться, что в нашем системе. присутствуют нужные устройства. Для этого наберем команду «**list disk**». Эта команда позволяет получить список дисков на вашем компьютере.



3. Выбираем тот диск, с которым будем работать при помощи команды «**select disk**». Указав значение своего диска. **select disk (порядковый номер диска).**

!!!Например: я буду использовать диск размером 232 Гбайта под номером 1, поэтому я пишу команду «select disk 1»

```
DISKPART> select disk 1

Выбран диск 1 .

DISKPART>
```

4. После выбора диска мы можем производить с ним разные операции. Теперь мы создадим раздел на жестком диске. Чтобы посмотреть какие разделы мы можем создать на данном томе введем команду «**create partition help**»

```
DISKPART> create partition help

Microsoft DiskPart, версия 10.0.19041.610

EFI         - Создание системного раздела EFI.
EXTENDED    - Создание расширенного раздела.
LOGICAL     - Создать логический диск.
MSR         - Создание резервного раздела Майкрософт.
PRIMARY     - Создание основного раздела.

DISKPART>
```

5. Перед созданием нового раздела нам нужно **освободить** место под новый раздел.

Для этого нам нужно сжать раздел на нашем ПК. Посмотрим какие у нас есть разделы на ПК. Введем команду «**lis vol**».

```
DISKPART> lis vol

  Том    ###  Имя  Метка        ФС     Тип         Размер   Состояние  Сведения
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
     Том 0     F   Данные       NTFS   Простой то   163 Gб  Исправен
     Том 1     E   Общие данны  NTFS   Простой то   767 Gб  Исправен
     Том 2     D                       DVD-ROM        0 б  Нет носит
     Том 3         Зарезервиро  NTFS   Раздел       579 Мб  Исправен   Системны
     Том 4     C                NTFS   Раздел       232 Gб  Исправен   Загрузоч
```

6. Выдерите тот раздел, который будите сжимать.

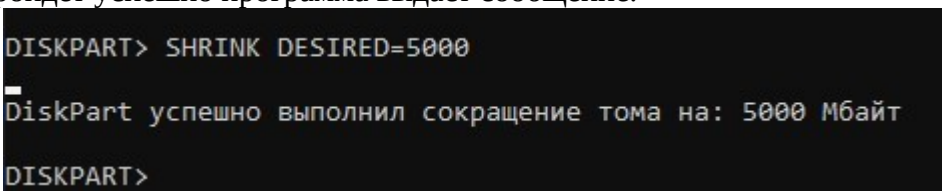
Например: я выбрал том под номером «4» (это диск С) и его буду сжимать.

```
DISKPART> lis vol

  Том    ###  Имя  Метка        ФС     Тип         Размер   Состояние  Сведения
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
     Том 0     F   Данные       NTFS   Простой то   163 Gб  Исправен
     Том 1     E   Общие данны  NTFS   Простой то   767 Gб  Исправен
     Том 2     D                       DVD-ROM        0 б  Нет носит
     Том 3         Зарезервиро  NTFS   Раздел       579 Мб  Исправен   Системны
     Том 4     C                NTFS   Раздел       232 Gб  Исправен   Загрузоч
```

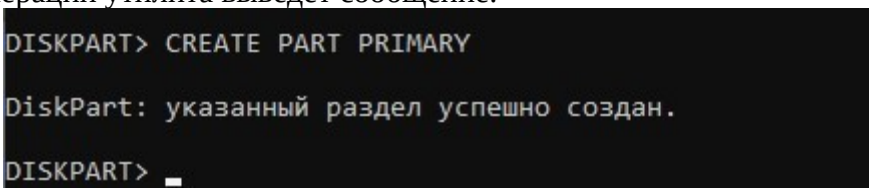
7. Выбираем том с помощью команды «**sel vol**» указам при этом номер тома. **Например: «sel vol 4»**

```
DISKPART> sel vol 1

Выбран том 1.

DISKPART>
```

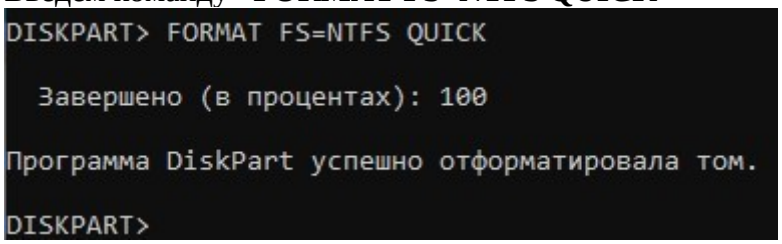
8. Мы будем сжимать диск на 5 ГБ. Для этого введем команду «**SHRINK DESIRED=»** после знака равенства задаем размер в мегабайтах, указываем 5000 (5000 МБ = 5 ГБ). Как только операция пройдет успешно программа выдаст сообщение.



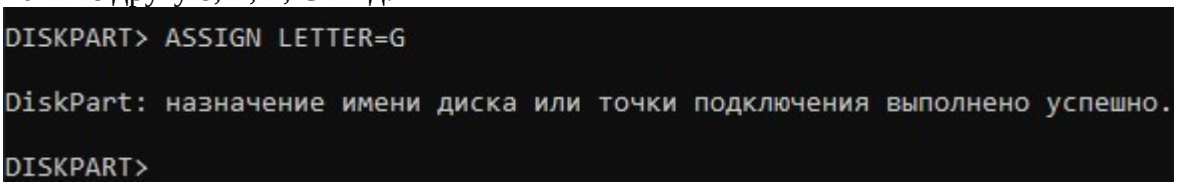
9. Теперь создадим новый раздел, для этого введем команду «**CREATE PART PRIMARY**». После успешной операции утилита выведет сообщение.



10 Теперь отформатируем созданный раздел назначив ему тип файловой системы **NTFS**. Введем команду «**FORMAT FS=NTFS QUICK»**



11. Далее назначим букву для тома. Введем команду «**ASSIGN LETTER=D**». Если буква D занята, то укажите другую, E, F, G и тд.



После присвоения буквы новый том будет готов, и система автоматически его откроет. Теперь вы можете пользоваться новым разделом!

12. Выйдите из утилиты набрав команду «**EXIT»**

Оптимизация и дефрагментация

Оптимизация — модификация системы для улучшения её эффективности. Система может быть одиночной компьютерной программой, цифровым устройством, набором компьютеров или даже целой сетью.

Хотя целью оптимизации является получение оптимальной системы, истинно оптимальная система в процессе оптимизации достигается далеко не всегда. Оптимизированная система обычно является оптимальной только для одной задачи или группы пользователей: где-то может быть важнее уменьшение времени, требуемого программе для выполнения работы, даже ценой потребления большего объёма памяти; в приложениях, где важнее память, могут выбираться более медленные алгоритмы с меньшими запросами к памяти.

Более того, зачастую не существует универсального решения (хорошо работающего во всех случаях), поэтому инженеры используют компромиссные (англ. tradeoff) решения для оптимизации только ключевых параметров. К тому же, усилия, требуемые для достижения полностью оптимальной программы, которую невозможно дальше улучшить, практически всегда превышают выгоду, которая может быть от этого получена, поэтому, как правило, процесс оптимизации завершается до того, как достигается полная оптимальность. К счастью, в большинстве случаев даже при этом достигаются заметные улучшения.

Дефрагментация — процесс перераспределения фрагментов файлов и логических структур файловых систем на дисках для обеспечения непрерывной последовательности кластеров.

В случае использования ***жёстких дисков***, вследствие дефрагментации ускоряется чтение и запись файлов, а следовательно, работа программ и операционной системы. Это достигается за счёт того, что чтение файлов после дефрагментации производится максимально линейно и непрерывно, без дополнительных передвижений головки жёсткого диска для поиска и воссоединения фрагментов.

Твердотельные накопители и флешки на основе различных видов ***флэш-памяти*** в дефрагментации файловых систем не нуждаются. Более того, дефрагментация, в некоторой степени, вредит флэш-памяти, так как последняя имеет намного более ограниченное количество циклов записи/перезаписи, чем накопители на магнитных дисках при должном использовании.

Фрагментация данных — разбиение файла на фрагменты различной величины для записи в различные (не последовательные) области жёсткого диска. Фрагментация применяется, как правило, в случае необходимости записи большого файла на диск, где недостаточно непрерывного объёма, но достаточно суммарного свободного места (например, вследствие удаления ранее записанных файлов).

Последствия

Когда фрагментированных файлов становится много, скорость работы накопителя (особенно жёсткого диска или дискеты) с фрагментированными файлами уменьшается, поскольку возникают задержки на физическое перемещение головки жёсткого диска между разделёнными фрагментами. Более того, общая фрагментация жёсткого диска замедляет не только скорость работы с уже существующими файлами, но и, как правило, существенно замедляет запись новых.

Задание №2:

Порядок работы

1. Создать в личной папке на жестком диске файл «Отчет о выполнении практической работы», в верхнем колонтитуле созданного документа укажите фамилию и имя.
2. Создать таблицу и заполнить ее:

1	Модель жесткого диска			
2	Имена логических дисков (томов)			
3	Файловая система			
4	Емкость			
5	Объем занятого места	до очистки диска		
		после очистки диска		
6	Объем свободного места	до очистки диска		
		после очистки диска		
7	Процент свободного места	до очистки диска		
		после очистки диска		
8	Размер кластера			
9	Всего файлов			
10	Средний размер файла			
11	Количество фрагментированных файлов	до дефрагментации		
		после дефрагментации		
12	Количество лишних фрагментов	до дефрагментации		
		после дефрагментации		
13	Всего фрагментировано, %	до дефрагментации		
		после дефрагментации		
14	Фрагментация файлов,%	до дефрагментации		
		после дефрагментации		

3. Открыть «Мой компьютер», определить количество логических дисков (томов) жесткого диска, внести в таблицу их имена.

4. Открыть контекстное меню к одному из логических дисков жёсткого диска, открыть пункт Свойства и вкладку Оборудование.
5. Внести в первую строку таблицы наименование модели жесткого диска.
6. Открыть программу «Дефрагментация диска». Для этого выполните: Пуск/ Программы/ Стандартные/ Служебные/Дефрагментация диска.
7. В открывшемся диалоговом окне выбрать информацию для заполнения строк таблицы с 3-й по 7-ю (в 5, 6, 7-й строках заполните позиции «до очистки диска»).
8. Получить информацию о фрагментации логических дисков (томов) жесткого диска. Для этого в диалоговом окне Дефрагментация диска для каждого тома выполните следующее:
9. выделить том и активизировать кнопку Анализ; в результате начнется процесс анализа фрагментированности диска;
10. по окончании анализа активизировать кнопку Вывести отчет;
11. по отчету об анализе заполнить оставшиеся строки таблицы (в строках с 11-й по 14-ю заполните позиции «до дефрагментации»).
12. Закрыть программу дефрагментации диска.
13. Провести поочередно очистку логических дисков (томов) жесткого диска; для этого выполните: Пуск/ Программы/ Стандартные/ Служебные/ Очистка диска. Вам предстоит выбрать диск и запустить программу очистки диска.
14. Открыть программу «Дефрагментация диска».
15. В открывшемся диалоговом окне выбрать информацию для заполнения позиций «после очистки диска» в 5, 6, 7-й строках таблицы.
16. Провести поочередно дефрагментацию логических дисков жесткого диска 17. Вывести отчет о дефрагментации.
18. По отчету о дефрагментации заполнить позиции «после дефрагментации» в строках с 11-й по 14-ю.

Задание №3:

Проверка диска: графический интерфейс

1. Открыть контекстное меню Компьютер, выбрать диск и выбрать Свойства.
2. На вкладке Сервис нажмите кнопку Выполнить проверку.
3. Выберите один из вариантов проверки:

- Чтобы проверить диск без попыток исправления ошибок в случае их обнаружения, снять оба флажка и нажать кнопку Запуск.
- Чтобы выполнить поиск ошибок файлов и папок и исправить их, установить флажок Автоматически исправлять системные ошибки и нажать кнопку Запуск.
- Чтобы проверить поверхность диска на наличие физически поврежденных (bad) секторов и попытаться восстановить хранящиеся в них данные, выбрать Проверять и восстанавливать поврежденные сектора и нажать кнопку Запуск.
- Чтобы выполнить проверку файловых и физических ошибок и попытаться исправить их, установить оба флажка и нажмите кнопку Запуск.
- Если выбрать Автоматически исправлять системные ошибки для используемого диска, будет предложено выполнить проверку диска в ходе следующей загрузки компьютера.
- Во избежание повреждения диска и хранящихся на нем данных, не прерывайте и не останавливайте начавшуюся проверку.

По окончании проверки на экран будут выведены её результаты.

Задание №4:

Проверка диска: командная строка Пояснение:

Синтаксис проверки диска:

CHKDSK [том[[путь]имя_файла]] [/F] [/V] [/R] [/X] [/I] [/C] [/L[:размер]] [/B]

Порядок работы

- В меню Пуск выбрать Выполнить;
- Ввести команду cmd, нажать Enter. Откроется окно DOS;

- Ввести команду chkdsk c: (где c: – проверяемый диск) и нажать Enter. Диск проверяется, и выдаются результаты проверки.
- Для закрытия окна ввести команду exit и нажать Enter.
- Если в команду chkdsk добавить параметр /f, то будет выдано предупреждение о невозможности проверки и предложение, задать проверку, при следующей загрузке Windows.

CHKDSK	Команда запускает проверку диска на наличие ошибок. Если ни один флаг не установлен, проверка осуществляется в режиме только чтения (если ошибки будут обнаружены, программа проверки диска не будет пытаться исправлять их).
Том	Укажите букву проверяемого диска с двоеточием. Например, CHKDSK C:
имя_файла	Название и расширение файла, который нужно проверить на наличие фрагментации (только для дисков с файловыми системами FAT и FAT32). Необходимо указать полный путь к файлу. Например, чтобы проверить фрагментацию файла wseven.txt, расположенного в папке «Windows» на флэш-диске G, введите CHKDSK G:\ WINDOWS\WSEVEN.TXT и нажмите Ввод.
/F	Исправление ошибок на диске. Например, чтобы проверить диск C и исправить ошибки в случае их обнаружения, введите CHKDSK C: /F и нажмите Enter.
/R	Поиск поврежденных секторов и восстановление хранящихся в них данных. Должен быть обязательно установлен флаг/ F. Например, чтобы проверить поверхность диска C на наличие физически поврежденных секторов и восстановить хранящиеся в них данные, введите CHKDSK C: /F /R и нажмите Enter.
/V	Если этот флаг установлен, во время проверки дисков с файловой системой FAT/FAT32 выводится полный путь и имя каждого файла на диске. Для дисков с файловой системой NTFS: вывод сообщений об очистке (при их наличии).
/X	Предварительное отключение тома (при необходимости). Все откры- тые дескрипторы для этого тома будут недействительны. Должен быть обязательно установлен флаг /F. Например, CHKDSK C: /F /X Флаги CHKDSK, действующие только во время проверки дисков с файловой системой NTFS
/L:размер	Этот флаг позволяет задать размер файла журнала (в килобайтах). Если размер не указан, выводится текущее значение размера. Например, чтобы узнать текущий размер файла журнала chkdsk для диска C, введите CHKDSK C: /L и нажмите Ввод. Чтобы проверить диск C, исправить системные ошибки на нем и задать новый размер файла журнала равный 80 мегабайтам, введите CHKDSK C: /F /L:81920 и нажмите Ввод. Обратите внимание, что для файла журнала требуется много места, и слишком маленькое значение установить не получится.

/I	Если этот флаг установлен, CHKDSK выполняется быстрее за счет менее строгой проверки элементов индекса.
/C	Если этот флаг установлен, CHKDSK пропускает проверку циклов внутри структуры папок.
/B	Если этот флаг установлен, CHKDSK сбрасывает ранее отмеченные поврежденные (bad) секторы и перепроверяет их. Должен быть обязательно установлен флаг /R. Например, чтобы проверить поверхность диска C на наличие физически поврежденных секторов с восстановлением хранящихся в них данных, а также перепроверить все секторы, отмеченные ранее как поврежденные, введите CHKDSK C: /F /R /B и нажмите Enter.

Контрольные вопросы:

1. Что такое форматирование диска?
2. Что такое диагностика диска?
3. Что такое оптимизация диска?
4. Что такое дефрагментация диска?
5. Для чего нужна очистка диска?

Практическая работа № 10

Сведения о системе. Сведения о восстановлении системы. Настройки Windows, особенности быстродействия системы

Цель работы:

- уметь настраивать операционную систему Windows для удобства работы с ней.

Краткие теоретические сведения:

Окно Панель управления представляет собой основное средство для проведения различных настроек компьютера. Значки, имеющиеся в этом окне, позволяют настраивать различные аппаратные и программные средства, а также задавать общие свойства системы

Открыть Панель управления можно несколькими способами:

1. Пуск - Настройка - Панель управления.
2. Дважды щелкнуть мышью по значку Мой компьютер, а затем по папке Панель управления.
3. Открыть Проводник и найти в нем папку Панель управления.

Задание 1. Настройка мыши

1. Дважды щелкнуть по значку Мышь на Панели управления.
2. Открыть вкладку Кнопки мыши (вкладка находится вверху окна, для ее открытия нужно щелкнуть по ней мышь).
3. Установить конфигурацию кнопок Для левши. Щелкнуть но кнопке ОК. Теперь значения кнопок поменялись и основной стала правая кнопка мыши.
4. Снова открыть Панель управления. Не забыть, что теперь щелкать нужно правой кнопкой мыши,
5. Изменить Конфигурацию кнопок на Обычную.
6. Скорость двойного нажатия используется для регулировки интервала времени при котором второй щелчок рассматривается как двойной. Установить скорость (перетаскивая бегунок по линейки). Дважды щелкнуть мышью в области проверки. Изменить скорость и снова щелкнуть в области проверки. Затем установить движок примерно посередине линейки и нажать ОК.
7. Щелкнуть по вкладке Перемещение. Задать произвольную скорость перемещения мыши и щелкнуть но квадратику Отображать шлейф. Нажать ОК. Переместить указатель мыши ипосмотреть на изменения. Затем все вернуть в исходное состояние.

Задание 2. Свойства мыши

1. Откройте диалоговое окно Свойства: Мышь (Пуск ► Настройка ► Панель управления ► Мышь).
2. Щелкните дважды в области проверки на панели **Скорость выполнения двойного щелчка**. Убедитесь, что при двойном щелчке элемент срабатывает, а при двух отдельных щелчках с продолжительным интервалом — нет.
3. Методом перетаскивания переместите движок **Скорость** в крайнее правое положение. Убедитесь, что при этом интервал времени между двумя отдельными щелчками, составляющими двойной щелчок, чрезмерно занижен и выполнить двойной щелчок очень трудно.
4. Переместите движок в крайнее левое положение и убедитесь в том, что два отдельных щелчка интерпретируются как двойной щелчок.
5. Экспериментально выберите наиболее удобное для себя положение движка.
1. Откройте вкладку Параметры указателя.
1. Уменьшите чувствительность мыши, переместив движок Задайте скорость движения указателя в крайнее левое положение. Щелкните на кнопке Применить.
2. Установите указатель мыши примерно в центре экрана. Не отрывая запястья от поверхности стола, подвигайте мышь в направлении влево-вниз — вправо-вверх. Убедитесь в том, что указатель мыши не достигает левого Нижнего и правого верхнего углов экрана.
3. Переместите движок Задайте скорость движения указателя в крайнее правое положение. Щелкните на кнопке Применить.
10. Убедитесь в том, что указатель мыши можно провести от левого нижнего до правого верхнего углов экрана, не отрывая запястья от поверхности стола.
11. Экспериментально выберите наиболее удобное для себя положение движка. После каждого изменения его положения не забывайте задействовать командную кнопку Применить. Оптимальный выбор может зависеть от конкретной модели мыши, наличия свободного места на поверхности стола и привычных навыков работы.
12. Закройте диалоговое окно Свойства: Мышь.

Задание 3. Настройка даты и времени

1. Щелкнуть по значку Дата и время на Панели управления.
2. В появившемся окне щелкнуть по вкладке Дата и время.
3. Посмотреть как можно изменить дату и время.
   ! **Ничего не изменять**

Задание 4. Просмотр шрифтов

1. Дважды щелкнуть по значку Шрифты на Панели управления.
2. Двойной щелчок по названию шрифта.
3. Просмотреть 5-6 различных шрифтов.
4. После просмотра шрифта окно закрыть.

Задание 5. Настройки фона рабочего стола

1. Дважды щелкнуть по значку Экран на Панели управления.
2. Щелкнуть по вкладке Фон.
3. Выбрать рисунок из списка Рисунок (например, Облака). Щелкнуть по кнопке ОК.
4. Переключатель Размножить позволяет размножать выбранный рисунок и покрыть рабочий стол копиями рисунка. Переключатель По центру позволяет разместить рисунок в центре рабочего стола.
5. Если рисунок не задан или расположен в центре рабочего стола, то поверхность рабочего стола можно заполнить узором, который выбирается в списке Фоновый узор.

6, Можно создать свой узор щелкнув по кнопке Изменить. Изменение узора выполнить с помощью мыши. Написать название вашего узора в поле Название Узор 1- и щелкнуть по кнопке Добавить. Для установки узора на Рабочий стол щелкнуть по кнопке ОК.

7. Вернуть фон Рабочего стола в исходное состояние.

Задание 6.Выбор и настройка экранной заставки

1. Дважды щелкнуть по значку Экран на Панели управления,
2. Щелкнуть по вкладке Заставка.
3. В поле Заставка выбрать любую заставку. Для просмотра заставки щелкнуть по кнопке Просмотр.
4. По окончании просмотра выбрать тип заставки - Нет. Щелкнуть по кнопке **ОК.**
5. Выберите в поле заставка Объемный текст. Нажмите кнопку Настройка Выберите пункт Текст (черная точка должна стоять в круге рядом со словом Текст). В поле справа введите номер своей группы Размер, разрешение, поверхность, скорость и стиль движения настройте по своему усмотрению Нажмите ОК.
6. Нажмите кнопку Просмотр. Просмотрите результат.

! **Никаких паролей не устанавливать!**

Задание 7. Настройка схемы оформления рабочего стола

1. Дважды щелкнуть по значку Экран на Панели управления.
2. Щелкнуть по вкладке Оформление.
3. Элемент оформления выбирается в списке Элемент. Выбрать Рабочий стол.
4. Выбрать в списке схему оформления Дождливый день. Щелкнуть по кнопке ОК.
5. Выбрать произвольную схему оформления.
6. По окончании просмотра выбрать схему оформления Стандартная Windows.

Задание 8.Изменение размера и положения Панели задач.

1. Изменить размер Панели задач: поместить указатель мыши на ее верхний край, чтобы он принял вид двунаправленной стрелки. Нажать левую кнопку мыши и не отпуская ее перетащить верхний край Панели задач вверх. Максимальная ширина Панели задач не может превышать половину экрана.
2. Вернуть Панель задач в исходное состояние.
3. Поместить Панель задач сбоку экрана: перетащить ее мышью.
4. Щелкнуть правой кнопкой мыши на Панели задач. В появившемся меню выбрать команду Свойства.
5. Установить флажок Автоматически убирать с экрана. Щелкнуть мышью по кнопке ОК.
6. Вернуть Панель задач в исходное состояние.

Задание 9. Настройка оформления Рабочего стола, работа с Проводником, поисковой системой Windows XP и Корзиной

1. Выберите в контекстном меню пункт Свойства — откроется диалоговое окно Свойства: Экран. Откройте вкладку Рабочий стол.
2. В списке Фоновый рисунок выберите рисунок Японский мотив. Щелкните на кнопке ОК. Убедитесь в том, что фон Рабочего стола изменился.
3. Повторите пункты 2-3, изменяя на вкладке Рабочий стол способ расположения фонового рисунка с помощью раскрывающегося списка Расположение. Установите, как влияют на оформление экрана способы По центру, Замостить и Растянуть,
4. Повторите пункты 2-3, выбрав в качестве фонового рисунка объект Безмятежность и способ расположения Растянуть.
1. Запустите программу Проводник (Пуск ► Программы > Проводник).
5. Из Проводника запустите поисковую систему *Windows XP* (Вид» Панели обозревателя » Поиск > Файлы и папки).

6. С помощью поисковой системы установите, где хранятся фоновые рисунки Рабочего стола. Для этого в поле Часть имени файла или имя файла целиком введите название объекта Японский мотив, в поле Поиск в выберите пункт Локальные диски. Убедитесь в том, что в разделе Дополнительные параметры установлены флажки Поиск в системных папках и Просмотреть вложенные папки. Запустите процесс поиска щелчком на командной кнопке Найти.
7. Когда объект Японский мотив будет найден, на панели результатов поиска будет показано его местоположение — папка \Windows.
8. Щелкните на имени найденного файла правой кнопкой мыши и выберите в контекстном меню команду Открыть содержащую объект папку. В открывшемся окне папки посмотрите, в каком формате хранится этот и другие фоновые рисунки и узоры для Рабочего стола. Закройте окно поиска.
9. Закройте все открытые окна.

Словарь новых терминов (записать в тетрадь).

Кнопка - представляет собой прямоугольник с надписью. Кнопка используется для выполнения таких команд, которые сопровождаются закрытием диалогового окна (Отмена), изменением его содержания (ОК) или открытием нового окна

Текстовое поле - представляет собой белый прямоугольник, как бы утопленный в окно Текстовое поле предназначено для ввода строки текста (например имени файла).

Список - представляет собой прямоугольную область, в которой в горизонтальных строках располагаются пункты списка- При необходимости справа располагается вертикальная полоса прокрутки (список шрифтов в редакторе WordPad).

Вкладка - это отдельная страница диалогового окна. Вкладка состоит из корешка, на котором написано название вкладки, и страницы на которой располагаются элементы управления.

Флажок представляет собой элемент управления, позволяющий запретить или разрешить определенное действие. Состоит из белого квадратика и подписи, описывающей назначение флажка. Если флажок установлен, то квадратик помечается галочкой и действие разрешается.

Переключатель - используется для выбора одной возможности из нескольких, включенный переключатель помечается черной точкой.

Критерии оценки:

- оценка «зачтено» выставляется студенту, если правильно выполнены все задания лабораторной работы;
- оценка «не зачтено» выставляется студенту, если неправильно выполнены некоторые задания или выполнены не все задания лабораторной работы.

Преподаватель: _____
(подпись)