

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Институт прикладной математики и компьютерных наук
Кафедра «Вычислительная техника»

Утверждено на заседании кафедры
«Вычислительная техника»
«26» января 2022 г., протокол № 7

Заведующий кафедрой

_____ А.Н. Ивутин

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению контрольно-курсовой работы
по дисциплине (модулю)**

«Операционные системы и среды»

**основной профессиональной образовательной программы
высшего образования – программы бакалавриата**

по направлению подготовки
09.03.01 Информатика и вычислительная техника

с направленностью (профилем)
Электронно-вычислительные машины, комплексы, системы и сети

Форма обучения: заочная

Идентификационный номер образовательной программы: 090301-02-22

Тула 2022 год

Разработчик(и):

Андриянова М.А., доцент каф. ВТ, доцент, канд. техн. наук

(ФИО, должность, ученая степень, ученое звание)

(подпись)

Часть 1

1.1. Сбор сведений о системе и управление процессами

Цель работы: знакомство с командами по работе с процессами и командами, предоставляющими сведения о системе.

Сбор сведений о системе

Объекты и средства исследования: Объекты исследования: **uname, who, df, ps, grep, pstree, top**. Средства исследования: операционная система Linux

Подготовка к работе

Изучить описание и синтаксис следующих команд: **uname, who, df, ps, grep, pstree, top**.

Программа работы:

Начальные условия: Командная строка после входа в систему.

1. Определить имя текущей UNIX-системы.
2. Вывести содержимое каталога **proc**. Примечание: Признаком операционной системы Linux является наличие каталога **/proc**, которая содержит псевдофайлы, предоставляющие системную информацию и позволяющие настраивать ядро.
3. Вывести текущие пользовательские сеансы.
4. Вывести список всех примонтированных устройств. Первый столбец — имя устройства, второй — точка монтирования, далее следует тип файловой системы и опции монтирования.
5. Вывести загруженность примонтированных дисков.
6. Вывести информацию о всех выполняющихся процессах.
7. Оставить в выводе предыдущей команды только системные процессы. Пояснения: Часть процессов, показанные в квадратных скобках, — это потоки ядра, исполняющие специальные функции. Можно выделить демонов — в столбце терминала они имеют символ «?». Например, системный журнал (**syslog**) или планировщик задач (**cron**).
8. Вывести иерархию процессов. Примечание: В вершине дерева стоит процесс **init**.
9. Рассмотреть поведение процессов интерактивно.

Управление процессами с помощью сигналов

Объекты и средства исследования: Объекты исследования: **find, aux, kill, killall**.

Средства исследования: операционная система Linux

Подготовка к работе

Изучить описание и синтаксис следующих команд: **find, aux, kill, killall**.

Программа работы:

Начальные условия: Командная строка после входа в систему.

1. Запустить команду **yes**, производящую бесконечный вывод символа **y** на экран. Прервать её нажатием **Ctrl-C**. При этом запущенному процессу будет отправлен сигнал **SIGTERM** — завершение программы.
2. Запустить сбор информации обо всех файлах системы.
3. Найти идентификатор запущенного только что процесса, запущенной в другом терминале. Первая строка содержит необходимый **PID**.
4. Отправить сигнал завершения этому процессу, указав идентификатор процесса. Убедитесь, что на терминале с запущенной командой появились строка «Terminated» и приглашение.
5. Попытаться завершить системный процесс. Вы получите сообщение об ошибке доступа. Отправление сигналов системным процессам может производить только суперпользователь.
6. Альтернативным способом отправления сигналов процессам — по имени процесса, а не по **PID** — является команда **killall**. Выполнить команду **killall bash**, и завершить

все командные оболочки, а тем самым и сеансы пользователей.

7. Сигнал завершения процесса может перехватываться и игнорироваться программами. Неперехватываемым является сигнал SIGKILL. Отправить сигнал SIGKILL.

Выполнение задач в фоновом режиме

Объекты и средства исследования: Объекты исследования: find, jobs, fg, bg. Средства исследования: операционная систем Linux

Подготовка к работе

Изучить описание и синтаксис следующих команд: find, jobs, fg, bg

Программа работы:

Начальные условия: Командная строка после входа в систему.

1. Запустить длительную команду, например `find / > files.txt` и приостановить её выполнение с помощью нажатия Ctrl-Z. При этом процессу посылается сигнал SIGSTOP. Команда приостановлена и запомнена как задача 1 (номер в квадратных скобках).
2. Посмотреть текущий список запущенных задач командной оболочки.
3. Возобновить исполнение задания.
4. Еще раз приостановить процесс и запустить задачу в фоновом режиме. Тогда можно будет продолжать работу в командной строке.

Команды можно сразу запускать в фоновом режиме. При этом необходимо добавить символ «&» (амперсанд) в конец строки команды: `find / -name "*.xml" > xml-list &`. При этом командная оболочка выводит номер задания и PID созданного процесса.

Запуск демонов

Объекты и средства исследования: Объекты исследования: find, nohup. Средства исследования: операционная систем Linux

Подготовка к работе

Изучить описание и синтаксис следующих команд: find, nohup.

Программа работы:

Начальные условия: Командная строка после входа в систему.

1. Запустить команду в фоновом режиме и выйти из командной оболочки: например, `find / -name "*.html" -exec grep -Hn "linux loader" \{} \; &` и `exit`. Запущенная команда завершится по сигналу SIGHUP.

Для того, чтобы программы не получали сигнал SIGHUP, используется специальная команда `nohup`: `nohup find / -name "*.txt" -exec grep -Hn "linux loader" \{} \; &`. Эта программа завершится корректно после окончания поиска.

Изменение приоритетов выполняющихся программ

Объекты и средства исследования: Объекты исследования: ps, nice, renice. Средства исследования: операционная систем Linux

Подготовка к работе

Изучить описание и синтаксис следующих команд: ps, nice, renice

Программа работы:

Начальные условия: Командная строка после входа в систему.

1. Определить уровни приоритетов процессов в системе (в UNIX они называются «nice»). По умолчанию, приоритет процессов простого пользователя равен нулю.
2. Запустить создание архива с пониженным приоритетом.
3. Понизить приоритет заранее запущенного процесса архивации. Простые пользователи не могут повышать приоритет процессов, только понижать.

1.2. Сервисные программы

1.Цель и задачи работы: Знакомство с утилитами и приобретение навыков работы с ними.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Внимание! В работе использовать средства администрирования MS Windows*

1. Проверить файловую систему любого логического диска одной из утилит.
2. Провести оптимизацию любого логического диска.
3. Скопировать необходимые программы в рабочий каталог.
4. Создать архивные файлы, используя все возможные степени сжатия программ-архиваторов.
6. Произвести тестирование полученных архивов.
7. Просмотреть содержимое любого архива, используя различные (возможные) критерии сортировки при выводе.
8. Произвести обновление одного (нескольких) файла (файлов) в архиве.
9. Произвести удаление нескольких файлов из архива и вывести каталог полученного архива.
10. Произвести извлечение нескольких файлов из архива всеми возможными способами (для данного архиватора).
11. Создать самораскрывающийся архив. Проверить его работоспособность.
12. Определить конфигурацию компьютера.
13. Все выполняемые этапы работы отразить в отчете.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое файловая система?
2. Как возникает фрагментация?
3. Какая бывает фрагментация?
4. Какие утилиты тестируют файловую систему?
5. Какие утилиты тестируют поверхность диска?
6. Что такое дефрагментация?
7. Какие режимы дефрагментации существуют? Чем они отличаются?
8. Что такое оптимизация магнитного носителя?
9. Для чего предназначены программы-архиваторы?
10. Что такое самораскрывающийся архив?
11. Зачем нужно создавать резервные копии данных?
12. Какие средства существуют для создания архивных копий?

1.3. Основные приемы работы в командной оболочке

Цель и задачи работы: Знакомство с интерактивными оболочками операционных систем и приобретение навыков работы с ними.

Оборудование

IBM совместимые ЭВМ, операционная среда WINDOWS, оболочки Norton Commander, Volcov Commander, FAR, Windows Commander, сервисные программы.

Программа работы

1. Настроить оконный интерфейс оболочки (в соответствии с вариантом)

Варианты настройки NC, VC, FAR и др.

Параметры	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1. Параметры окон:															
- левое	+		+	+		+	+		+	+		+	+		+
- правое		+	+		+	+		+	+		+	+		+	+
- вид информации:															
Brief	+	+	+L							+	+				+R
Full				+	+	+L			+R				+	+	
Info			+R			+R			+L			+R			+L
Tree							+	+				+L			
- сортировка:															
по именам	+				+				+				+		
по расширениям		+				+				+				+	
по дате создания			+				+				+				+
по размерам				+				+				+			
2. Параметры экрана:															
- размером окна															
полное	+		+		+		+		+		+		+		+
сокращенное		+		+		+		+		+		+		+	
- цвет экрана:															
Black & White		+		+		+		+		+		+		+	
Color	+		+		+		+		+		+		+		+
- статусная строка	+		+		+		+		+		+		+		+
- задержка до момента гашения экрана															
5 minutes	+			+			+			+			+		
3 minutes		+			+			+			+			+	
1 minute			+			+			+			+			+
3. Другие параметры:															
- автоматический вызов NC.MNU		+		+		+		+		+		+		+	
- путь:															
полный	+		+		+		+		+		+		+		+
краткий		+		+		+		+		+		+		+	
- строка функциональных клавиш	+		+		+		+		+		+		+		+
- верхняя строка меню		+		+		+		+		+		+		+	
- часы	+		+		+		+		+		+		+		+
- автоматическое сохранение последних параметров настройки	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Примечание: + - включение режима;
+L - для левого окна;
+R - для правого окна.

2. Создать собственный подкаталог и войти в него.

3. Создать три текстовых файла с расширением txt.
4. Переименовать любой из созданных файлов.
5. Просмотреть на экране дисплея содержимое созданных файлов.
6. Объединить все созданные файлы.
7. Создать еще один подкаталог и скопировать в него ранее созданный файл, полученный в результате слияния.
8. Выйти в корневой каталог и провести поиск файла и просмотр дерева.
9. Удалить созданные текстовые файлы и подкаталоги.
10. Все выполняемые этапы работы отразить в отчете.

Оформление отчета

Отчет должен содержать

- название и цель каждой части,
- последовательность команд со скриншотами полученных результатов и необходимыми пояснениями к п. 1.1,
- список используемых утилит с их назначением и результатами работы, а также порядок действий по выполнению программы работы к п. 1.2,
- номер варианта, скриншот результата настройки, скриншоты последующих действий с пояснениями к п.1.3.

Контрольные вопросы

1. Какова структура окон различных оболочек?
2. Что такое статусная строка?
3. Какие команды можно выполнить с помощью функциональных клавиш?
4. Каким образом вызывается верхнее меню в NC, VC, FAR?
5. Чем отличаются режимы Brief и Full?
6. Каким образом можно быстро найти нужный файл?
7. Что такое файл меню пользователя?
8. Каковы правила работы с меню пользователя?
9. Каковы правила написания файла меню пользователя?
10. Что такое файл расширений?
11. Для чего нужен файл ассоциаций?
12. Каковы правила написания файла ассоциаций?
13. Каковы особенности файла ассоциаций в Windows?
14. Как настроить оболочки?
15. Что такое тегирование? Как выполняются операции над группой файлов?
16. Чем отличаются режимы F3 и F4?
17. Чем отличаются режимы F5 и F6?
18. Как создать новый текстовый файл?
19. Для чего нужна клавиша F9?
20. Как удалить каталог?

Часть 2

2.1. ФАЙЛОВАЯ СИСТЕМА MS DOS

1. Цель и задачи работы: Научиться создавать файлы и читать записанную в них информацию с помощью функций DOS.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1. Основные характеристики файловой системы MS-DOS

В машинах типа IBM PC предусмотрены два уровня обращения к магнитным дискам. При работе на нижнем уровне пользователь с помощью прерывания BIOS INT 13h обращается непосредственно к драйверу диска. Типичными операциями этого уровня являются запись или чтение сектора, позиционирование головок, форматирование дорожки. Файловая система DOS не используется; требуемая информация отыскивается не по имени файла, а по номерам поверхности, цилиндра и сектора.

Верхний уровень реализуется с помощью прерывания DOS INT 21h, поддерживающего, наряду с прочими, также и функции обслуживания файловой структуры. Программист работает не с драйвером диска, а с файловой системой DOS, получая возможность оперировать такими понятиями файловой системы, как логический диск, каталог, файл.

Как известно, для удобства работы с большим количеством разнородных файлов в DOS используется древовидная структура каталогов. Каталог представляет собой файл обычно относительно небольшого размера, в котором содержится перечень всех подкаталогов следующего уровня и файлов, входящих в данный каталог. Каждому подкаталогу или файлу отводится одна запись размером 32 байт, в которую DOS заносит информацию о файле: имя, начальный адрес на цилиндре и треке (номер кластера), дата и время создания, длина в байтах, а также набор характеристик файла, называемых его атрибутами. Кроме записей, относящихся к нижележащим каталогам и файлам, каждый каталог содержит еще две записи: о себе самом и о родительском каталоге.

Атрибуты файла (в том числе файла каталога) хранятся в специально отведенном для них байте атрибутов и имеют следующие значения:

- 01h - файл только для чтения. Модификация или удаление файла запрещаются DOS;
- 02h - скрытый файл (не "замечаемый" командой DIR и некоторыми функциями DOS);
- 04h - системный файл (обычно с программами самой DOS);
- 08h - запись о файле представляет собой метку тома. Такая запись (одна на весь том) может существовать только в корневом каталоге;
- 10h - файл представляет собой каталог;
- 20h - файл после создания или модификации не был архивирован и, следовательно, нуждается в архивации (атрибут архивации).

При создании нового файла DOS сама отыскивает на диске свободное место и назначает его новому файлу, создавая и заполняя соответствующую этому файлу запись в каталоге. Хотя минимальной порцией информации, передаваемой контроллером диска в процессе записи или чтения файла, является сектор (512 байт) и программы BIOS работают как раз с секторами, файловая система назначает место на диске целыми кластерами. Размер кластера на гибком диске составляет два сектора (1 Кбайт); на жестком диске в кластер могут входить 4 - 8 и более секторов. Таким образом, минимальный физический размер файла, даже если данные в нем занимают лишь несколько байт, составляет один кластер. Однако в записи каталога указывается не физическая, а логическая длина файла, т.е. объем содержащихся в нем данных в байтах.

Методика работы с файлами существенно определяется тем обстоятельством, что каждый файл может занимать на диске несколько несмежных областей, т.е. быть

разрывным. Такая система выделения дискового пространства позволяет, во-первых, в процессе работы с файлом многократно дописывать в него новые данные, увеличивая при этом длину файла и, во-вторых, снимает проблемы с фрагментацией диска, поскольку даже самые маленькие и разрозненные свободные области на диске могут быть использованы для размещения нового файла. Следует, однако, иметь в виду, что сильно фрагментированный файл требует заметно больше времени для чтения или записи, что снижает скорость выполнения программ.

Процедура обращения к файлу в общем случае распадается на следующие операции:

- создание файла с заданным именем в указанном каталоге или открытие файла, если он был создан ранее;
- запись в файл или чтение из файла всего содержимого либо любой его части;
- закрытие файла.

Кроме перечисленных, имеется еще целый ряд вспомогательных операций: удаление файла, получение или установка его атрибутов, получение или изменение даты и времени создания файла, поиск файла с заданным именем (либо файлов, удовлетворяющих условиям заданного шаблона групповой операции) и др.

Существуют два способа выполнения операций с файлами: с использованием блоков управления файлами (FCB, File control block) и дескрипторов файлов (handle).

Использование FCB было характерно для первых версий DOS (до 2.0). FCB-структуры и соответствующие функции DOS не поддерживают древовидных каталогов, и поэтому они могут работать только с файлами из текущего каталога текущего диска. С их помощью, естественно, нельзя создавать или удалять сами каталоги. В настоящее время функции типа FCB остались в давно созданных программах и в данном пособии рассматриваться не будут.

Другой способ операций с файлами предполагает использование дескрипторов (файловых индексов, файловых описателей), которые представляют собой номера системных областей, располагаемых в оперативной памяти и закрепляемых за открываемыми файлами. Обращение к файлу (а также и к стандартному периферийному устройству - терминалу, принтеру и т.д.) осуществляется по номеру дескриптора, идентифицирующего данный файл или устройство. Дескриптор назначается файлу системой при его открытии и освобождается при закрытии файла.

Для работы со стандартными устройствами DOS предоставляет пять предопределенных дескрипторов:

- 0 - стандартный ввод (CON);
- 1 - стандартный вывод (CON);
- 2 - стандартная ошибка (CON);
- 3 - стандартный вспомогательный последовательный порт (AUX);
- 4 - стандартный принтер (PRN).

Таким образом, при работе с терминалом, принтером или последовательным портом нет необходимости открывать новые дескрипторы; ввод с клавиатуры осуществляется через дескриптор 0, вывод на экран - через дескрипторы 1 или 2, вывод на принтер - через дескриптор 4. Различие дескрипторов 1 и 2 заключается в том, что стандартный вывод (как и стандартный ввод) можно перенаправить средствами DOS на любое устройство или в файл, а стандартная ошибка всегда связана с экраном. Обычно дескриптор 2 используют для вывода на экран аварийных или диагностических сообщений.

Для облегчения ориентации в многочисленных функциях DOS, осуществляющих операции над файлами, каталогами и дисками, их удобно разбить на смысловые группы.

А. Создание, открытие и закрытие файла:

- 3Ch - создать файл;
- 5Ah - создать временный файл;
- 5Bh - создать новый файл;

3Dh - открыть файл;
3Eh - закрыть файл;
68h - сбросить файл на диск;
41h - удалить файл.

В. Запись и чтение данных:

42h - установить указатель;
3Fh - читать из файла или устройства;
40h - записать в файл или устройство.

С. Изменение характеристик файла:

43h - получить или установить атрибуты файла;
56h - переименовать файл;
57h - получить или установить дату и время создания файла.

Д. Поиск файла:

IAh - установить адрес области передачи данных (DTA);
2Fh - получить адрес области передачи данных (DTA);
4Eh - найти первый файл;
4Fh - найти следующий файл.

Е. Операции над каталогами:

39h - создать каталог;
3Ah - удалить каталог;
3Bh - сменить текущий каталог;
47h - получить текущий каталог.

Ф. Операции над дисками:

19h - получить текущий диск;
OEh - сменить текущий диск;
36h - получить информацию о диске.

2.2. Функции создания и чтения файла

Функции 3Ch и 5Bh позволяют создать файл с заданной спецификацией. Спецификация файла, т.е. путь к нему вместе с именем файла и расширением указывается в виде символьной строки, завершающейся двоичным нулем ("строки ASCIIZ"). Адрес этой строки заносится в регистры DS:DX. В регистре CX задается код атрибутов создаваемого файла: 0 - отсутствие атрибутов, 1 - только для чтения, 2 - скрытый, 4 - системный, 8 - метка тома, 20h - атрибут архива. Таким образом, с помощью этих функций можно создать как "настоящий" файл, так и метку тома (в корневом каталоге диска). В регистре AX возвращается дескриптор созданного файла, которым можно в дальнейшем пользоваться для записи в файл или чтения из него. Различие функций 3Ch и 5Bh проявляется лишь в случае, когда файл с заданной спецификацией уже существует. Функция 3Ch при этом фактически уничтожает имеющийся файл и создает новый с тем же именем, а функция 5Bh завершается с CF=1.

Функция 5Ah используется для создания временного файла, имя которому (являющееся функцией текущего времени) дает система. В регистрах DS:DX указывается адрес пути к файлу (не имени файла!), заданного в виде строки ASCIIZ, в конце которой должны быть предусмотрены 13 пустых байт, куда DOS поместит обратный слеш и имя создаваемого файла, завершаемое двоичным нулем. При необходимости файлу можно придать любые атрибуты (см. описание функций 3Ch и 5Bh) кроме атрибута метки тома. Обычно временные файлы удаляются перед завершением программы, причем забота об этом лежит на программисте (автоматически файл не удаляется). Для записи в созданный временный файл следует использовать дескриптор, возвращаемый функцией 5Ah в регистре AX.

Функция 3Dh позволяет открыть уже имеющийся файл. В регистрах DS:DX задается спецификация файла (путь и имя файла с расширением) в виде строки ASCIIZ; в регистре

AL - режим доступа (0 - чтение, 1 - запись, 2 - чтение и запись). В дальнейшем запись в файл и чтение из него осуществляется с помощью дескриптора, возвращаемого функцией в регистре AX.

Для каждого открытого файла DOS создает и поддерживает указатель, который представляет собой относительный номер байта в файле, начиная от которого будут выполняться запись и чтение данных. Указатель только что открытого или созданного файла позиционируется системой на начало файла, а функции чтения или записи смещают его на число прочитанных или записанных байт. Таким образом, повторное использование функций чтения или записи реализует последовательный доступ к файлу.

Функция 3Fh используется для чтения из файла или устройства. Перед вызовом функции в регистр BX помещается дескриптор, в регистр CX - число читаемых или записываемых байт, а в регистры DS:DX - адрес буфера в программе пользователя.

2.3. Примеры по программированию операций над файлами

Пример 1. Создание файла. В текущем каталоге диска создать файл с именем MYFILE.001.

```
;Определения
cr      equ    0Dh          ;Возврат каретки
lf      equ    0Ah          ;Перевод строки
text    segment 'code'
assume  CS:text, DS:data
myproc  proc
mov     AX, data
mov     DS, AX
;Создадим файл
mov     AH, 3Ch              ;Функция создания файла
mov     CX, 0                ;Без атрибутов
mov     DX, offset filename ;Адрес имени файла
int     21h
mov     handle, AX           ;Сохраним дескриптор файла
;Закроем файл (нет необходимости, если файл не надо читать повторно)
mov     AH, 3Eh              ;Функция закрытия
mov     BX, handle           ;Дескриптор
int     21h
;Завершим программу
outprog: mov     AX, 4C00h    ;Функция завершения, код
int     21h                  ;завершения = 0
myproc  endp
text    ends

data    segment
handle  dw      ?            ;Ячейка для дескриптора
filename db     'MYFILE.001', 0 ;Имя файла в формате ASCIIZ
data    ends

stack   segment para stack 'STACK'
db      128 dup (?)
stack   ends
end      myproc
```

Пример 2. Запись строки в файл.

;Основные фрагменты программы

```

;Создадим файл
...
;Запишем строку в файл
mov  AH,40h                ;Функция записи
mov  BX, handle             ;Дескриптор
mov  CX, stringln           ;Длина строки
mov  DX, offset string      ;Адрес строки
int  21h
;Закроем файл
...
;Завершим программу
...
;Поля данных
string db  'Текстовая строка',cr,lf  ;Строка для записи в файл
stringln equ  $-string              ;Ее длина
handle    dw  ?                     ;Ячейка для дескриптора

```

Пример 3. Чтение файла. Прочитать содержимое файла MYFILE.001 в память и вывести его на экран. Предполагается, что размер файла не более 80 байт.

```

;Определения
stdout equ 1                ;Дескриптор стандартного вывода
;Основные фрагменты программы
;Откроем файл
mov  AH,3Dh                ;Функция открытия файла
mov  AL,2                  ;Доступ для чтения/записи
mov  DX, offset filename   ;Адрес имени файла
int  21h
mov  handle,AX             ;Получили дескриптор
;Попытаемся прочитать 80 байт
mov  AH,3Fh                ;Функция чтения
mov  BX,handle             ;Дескриптор
mov  CX,80                 ;Столько читать
mov  DX, offset bufin      ;Сюда
int  21h
mov  CX,AX                 ;Столько реально прочитали
;Выведем прочитанное на экран
mov  AH,4Dh                ;Функция записи
mov  BX, stdout            ;Дескриптор стандартного вывода
mov  DX, offset bufin      ;Отсюда выводить (CX байт)
int  21h
;Завершим программу
...
;Поля данных
bufin    db  80 dup (' ')    ;Буфер ввода
handle    dw  ?              ;Ячейка для дескриптора
filename  db  'MYFILE.001',0 ;Имя файла

```

Пример 4. Изменение атрибутов файла. Установить у файла MYFILE.001 атрибут "только для чтения".

```

;Основные фрагменты программы
;Установим атрибут "только для чтения"
mov  AH,43h                ;Функция работы с атрибутами

```

```

mov AL,1           ;Установка атрибутов
mov CX,1           ;"Только для чтения"
mov DX, offset filename ;Адрес имени файла
int 21h

```

2.4. Функции прямого доступа к файлу

Для каждого открытого файла DOS создает и поддерживает указатель, который представляет собой относительный номер байта в файле, начиная от которого будут выполняться запись и чтение данных. Указатель только что открытого или созданного файла позиционируется системой на начало файла, а функции чтения или записи смещают его на число прочитанных или записанных байт. Таким образом, повторное использование функций чтения или записи реализует последовательный доступ к файлу.

Для организации прямого доступа к произвольному месту файла предусмотрена функция 42h, позволяющая задать положение указателя относительно начала файла (для этого надо задать AL=0), конца файла (AL=2) или текущего положения указателя (AL=1). Само значение смещения указателя (со знаком) заносится в регистры CX (старшая половина) и DX (младшая). Функция 40h используется для записи в файл или устройство. Перед вызовом функции в регистр BX помещается дескриптор, в регистр CX - число читаемых или записываемых байт, а в регистры DS:DX - адрес буфера в программе пользователя.

2.5. Примеры по программированию операций над файлами

Пример 5. Прямой доступ к файлу. Прочитать 8 байт из созданного ранее файла MYFILE.001, начиная с байта 5, вывести их на экран.

```

;Определения
stdout      equ 1           ;Дескриптор стандартного вывода
;Основные фрагменты программы
;Откроем файл
...
;Установим указатель
        mov AH,42h          ;Функция установки указателя
        mov AL,0            ;От начала файла
        mov BX, handle      ;Дескриптор
        mov CX,0            ;Старшая половина указателя
        mov DX,5            ;Младшая половина указателя
        int 21h

;Прочитаем 8 байт данных с помощью функции 3Fh
...
;Выведем прочитанное на экран с помощью функции 40h
...
;Завершим программу
...
;Поля данных
bufin      db 80 dup (' ')  ;Буфер ввода
handle     dw ?             ;Ячейка для дескриптора
filename   db 'MYFILE.001',0 ;Имя файла в формате ASCIIZ

```

Пример 6. Добавление данных к файлу. Добавить символьную строку к концу символьного файла.

```

;Основные фрагменты программы
;Откроем файл с указанным именем функцией 3Dh и сохраним
;полученный дескриптор в ячейке handle

```

```

...
;Установим указатель на конец файла
    mov     AH,42h           ;Функция установки указателя
    mov     AL,02           ;От конца файла
    mov     BX, handle       ;Дескриптор
    mov     CX,0             ;Старшая половина указателя
    mov     DX,0             ;Младшая половина указателя
    int     21h
;Допишем новую строку
    mov     AH,40h           ;Функция записи
    mov     BX, handle       ;Дескриптор
    mov     CX, stringln     ;Длина строки
    mov     DX, offset string ;Адрес строки
    int     21h
;Завершим программу
...
;Поля данных
string db    'Новая строка',cr,lf ;Строка для записи в файл
stringln equ  $-string           ;Ее длина
handle      dw    ?               ;Ячейка для дескриптора
filename    db    'MYFILE.001',0 ;Имя файла в формате ASCIIZ

```

Пример 7. Изменение характеристик файла. Изменить дату и время создания файла MYFILE.001.

```

;Основные фрагменты программы

;Откроем файл с указанным именем функцией 3Dh и сохраним
;полученный дескриптор в ячейке handle
...
;Изменим дату и время создания файла
    mov     AH,57h           ;Функция даты/времени
    mov     AL,1             ;Установить дату/время
    mov     BX, handle       ;Дескриптор файла
    mov     CX,0             ;Очистим CX
    or      CX, sec          ;Добавим секунды
    or      CX, min          ;Добавим минуты
    or      CX, hour         ;Добавим часы
    xor     DX, DX           ;Очистим DX
    or      DX, day          ;Добавим день
    or      DX, mon          ;Добавим месяц
    or      DX, year         ;Добавим год
    int     21h
;Завершим программу
...
;Поля данных
filename db    'MYFILE.001',0 ;Имя файла
handle   dw    ?               ;Ячейка для дескриптора
sec       dw    6/2            ;6 секунд
min       dw    15*32          ;15 минут
hour      dw    16*2048        ;16 часов
day       dw    25             ;25 число
mon       dw    3*32           ;март

```

year dw 13*512 ;13 лет от 1980 года

Пример 8. Переименование файла. Переименовать файл MYFILE.001, находящийся в текущем каталоге, дав ему имя NEWNAME.DAT.

;Основные фрагменты программы

;Настроим сегментный регистр ES на наш сегмент данных

push DS

pop ES

;Переименуем файл

mov AH,56h ;Функция переименования

mov DX, offset oldname ;Адрес старого имени

mov DI, offset newname ;Адрес нового имени

int 21h

;Завершим программу

...

;Поля данных

oldname db 'MYFILE.001',0 ;Старое имя файла

newname db 'NEWNAME.DAT',0 ;Новое имя файла

Пример 9. Пересылка файла в другой каталог на том же диске. Переслать файл NEWNAME.DAT из текущего каталога в нижележащий каталог NEWDIR, изменив при этом имя файла на NEWNAME.LEX.

;Основные фрагменты программы

;Настроим сегментный регистр ES на наш сегмент данных

...

;Переименуем файл

...

;Завершим программу

...

;Поля данных

oldname db 'NEWNAME.DAT',0 ;Старое имя файла

newname db 'NEWDIR\NEWNAME.LEX',0 ;Новое имя файла

Пример 10. Создание и удаление каталогов. Создать в текущем каталоге подкаталог NEWDIR. Удалить из текущего каталога подкаталог OLDDIR (обязательно пустой).

;Основные фрагменты программы

;Создадим новый каталог

mov AH,39h ;Функция создания каталога

mov DX, offset newname ;Адрес нового имени

int 21h

;Удалим ненужный каталог

mov AH,3Ah ;Функция удаления каталога

mov DX, offset oldname ;Адрес старого имени

int 21h

;Завершим программу

...

;Поля данных

oldname db 'OLDDIR',0 ;Старый каталог

newname db 'NEWDIR',0 ;Новый каталог

Пример 11. Смена диска. Сделать текущим диск A.

;Основные фрагменты программы

```

;Сделаем текущим диск A
      mov  AH,0Eh          ;Функция смены диска
      mov  DL,0            ;Код диска A (A=0, B=1, C=2 и т.д.)
      int  21h
;Завершим программу
...

```

ОБОРУДОВАНИЕ

IBM совместимые ЭВМ, операционная система MS DOS, модули ассемблера (masm и link), отладчик debug или Turbo Assembler.

ЗАДАНИЕ НА РАБОТУ

1. Ознакомиться с теоретическими положениями.
 2. Ответить на контрольные вопросы.
 3. Написать и отладить программу, которая выполняет следующие действия:
 - создает файл, содержащий заданное количество строк,
 - читает содержимое всего файла и выводит его на экран,
 - выводит на экран заданное количество байт из определенной строки.
- Варианты заданий.

	количество строк в файле	читаемые байты	строка
1	2 строки	10-15 и 2-5	1 строка
2	3 строки	2-7 и 9-12	3 строка
3	4 строки	11-15 и 3-5	2 строка
4	2 строки	2-6 и 9-14	1 строка
5	3 строки	13-18 и 4-5	2 строка
6	4 строки	2-3 и 9-19	4 строка
7	2 строки	11-15 и 6-9	2 строка
8	3 строки	2-4 и 9-16	1 строка
9	4 строки	12-15 и 4-5	3 строка
10	2 строки	2-9 и 5-12	2 строка
11	3 строки	10-15 и 2-10	2 строка
12	4 строки	4-7 и 9-16	4 строка

4. В соответствии с вариантом написать программу, которая добавляет к существующему файлу несколько строк и выводит его содержимое на экран. Добавление строк и вывод содержимого оформить в виде двух подпрограмм, включенных в текст основной программы (в виде отдельных процедур или макроопределений).

Варианты заданий.

	кол-во строк		кол-во строк		кол-во строк		кол-во строк
1	2 строки	4	3 строки	7	2 строки	10	2 строки
2	1 строку	5	1 строку	8	3 строки	11	1 строку
3	3 строки	6	2 строки	9	1 строку	12	3 строки

5. В соответствии с вариантом написать программу, которая создает новый каталог (при необходимости старый удалить), пересылает в него файл и изменяет его атрибуты.

Варианты заданий.

	вид обработки файла	старый каталог	установить биты	дополнительно изменить
1	пересылка	оставить	скрытый	дату создания
2	переименование	оставить	только для чтения	время создания
3	пересылка	удалить	системный	время создания
4	переименование	удалить	архивный	дату создания

	вид обработки файла	старый каталог	установить биты	дополнительно изменить
5	пересылка	оставить	только для чтения	дату создания
6	переименование	оставить	скрытый	время создания
7	пересылка	удалить	архивный	время создания
8	переименование	удалить	системный	дату создания
9	пересылка	оставить	системный	дату создания
10	переименование	оставить	архивный	время создания
11	пересылка	удалить	только для чтения	время создания
12	переименование	удалить	скрытый	дату создания

ПОРЯДОК РАБОТЫ

1. Написать и отладить программу в соответствии с полученным вариантом.
2. Оформить отчет.

ОФОРМЛЕНИЕ ОТЧЕТА

Отчет должен содержать вариант задания, программу и результаты ее выполнения.

КОНТРОЛЬНЫЕ ВОПРОСЫ

- 7.1. Какие два уровня обращения к дискам предусмотрены в машинах типа IBM PC?
2. Что представляет собой каталог?
3. Какие значения имеют атрибуты файла?
4. Каков минимальный физический размер файла?
5. Какая длина файла указывается в записи каталога?
6. Каков алгоритм обращения к файлу?
7. Что такое "блок управления файлами"?
8. Что такое "дескриптор"?
9. Перечислите предопределенные дескрипторы для работы со стандартными устройствами.
10. На какие смысловые группы можно разбить функции DOS?
11. Чем отличаются функции 3Ch, 5Bh и 5Ah?
12. Какая функция позволяет открыть уже созданный файл?
13. Как прочитать содержимое файла?
14. Как организовать прямой доступ к произвольному месту файла?
15. Что нужно сделать, чтобы добавить строку в конец файла?
16. Как можно изменить характеристики файла?
17. Как можно переименовать файл?
18. Как создать новый каталог?
19. Как удалить каталог?
20. Как переслать файл в другой каталог?
21. Как определить текущий диск?
22. Как сменить текущий диск?

2.2. Использование библиотеки макроопределений

1. Цель и задачи работы: Целью и задачами работы является приобретение навыков составления и использования библиотек макроопределений при программировании на языке Ассемблера.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1. Макроопределения

Макрокоманда представляет собой текстовую подстановку, в ходе выполнения которой каждый идентификатор определенного вида заменяется на цепочку символов из некоторого хранилища данных. Процесс выполнения макрокоманды называется макрогенерацией, а цепочка символов, получаемая в результате выполнения макрокоманды, — макрорасширением.

Процесс выполнения макрокоманд заключается в последовательном просмотре текста исходной программы, обнаружении в нем определенных идентификаторов и их замене на соответствующие строки символов. Причем выполняется именно текстовая замена одной цепочки символов (идентификатора) на другую цепочку символов (строку). Такая замена называется макроподстановкой.

Для того чтобы указать, какие идентификаторы на какие строки необходимо заменять, служат макроопределения. Макроопределения представляют собой последовательность операторов на языке Ассемблера, которые могут несколько раз появиться в программе. Тем самым они очень похожи на подпрограммы.

Макроопределения присутствуют непосредственно в тексте исходной программы. Они выделяются специальными ключевыми словами либо разделителями, которые не могут встречаться нигде больше в тексте программы. В процессе обработки все макроопределения полностью исключаются из текста входной программы, а содержащаяся в них информация запоминается для обработки при выполнении макрокоманд.

Например, пусть необходимо сложить содержимое двух ячеек памяти размером в слово, используя макроопределение:

```
; макроопределение для сложения двух значений
ADD_WORDS    MACRO    TERM1, TERM2, SUM
MOV          AX, TERM1
ADD          AX, TERM2
MOV          SUM, AX
ENDM
```

Тогда в программе можно использовать оператор:

```
ADD_WORDS PRICE, TAX, COST
```

Вместо этого оператора Ассемблер после трансляции вставит в программу следующие команды:

```
MOV AX, PRICE
ADD AX, TAX
MOV COST, AX
```

Макроопределения при написании программы на языке ассемблера не включаются ни в один сегмент. Их описывают перед сегментами после заголовка. Структура программы, содержащей макроопределения, может иметь следующий вид:

```
title  ex_prog    - заголовок программы
        page      ,132
имя1 macro [параметры] - первое макроопределение
...
endm
...
имяN macro [параметры] - N-ое макроопределение
...
endm
stack segment para stack 'stack'    - сегмент стека
```

```

...
stack ends
dseg segment para public 'data'      - сегмент данных
...
dseg ends
cseg segment para public 'code'      - сегмент команд
...
cseg ends
      end our_prog - конец программы

```

2.2. Библиотека макроопределений.

Библиотека макроопределений представляет собой файл с расширением lib, в котором заданы макроопределения, необходимые для нескольких программ. Этот файл можно создать с помощью любого текстового редактора точно так же, как и любую программу. После того как такой файл создан, его содержимое можно считывать в любую исходную программу. Тем самым все макроопределения библиотеки становятся доступными для этой программы. Чтобы использовать какой-либо из них, достаточно указать его имя.

Включаемые в библиотеку макроопределения должны быть процедурами общего пользования, которыми можно воспользоваться практически в любой программе. Поэтому, создавая библиотеку макроопределений, придерживайтесь следующих указаний:

- Старайтесь делать макроопределения как можно более универсальными.
- Если макроопределение содержит метки, перечислите их в операторе LOCAL.
- Сохраняйте все используемые в макроопределении регистры, за исключением выходных, с помощью операторов PUSH и POP.

2.3. Псевдооператоры Макроассемблера.

Псевдооператоры общего назначения

MACRO - присваивает ИМЯ последовательности операторов языка Ассемблера.

Формат:

имя MACRO [список формальных параметров]

...

ENDM

LOCAL - должен предшествовать любому другому оператору в макроопределении и даже комментариям. Указание метки в этом операторе означает, что такую же метку можно использовать в других макроопределениях. Ассемблер дает ей новое имя всякий раз, когда расширяет макроопределение; дублирование меток исключается. Формат:

LOCAL [список формальных параметров]

Условный псевдооператор

IF1 - выполняет, если Ассемблер осуществляет первый проход. Формат:

IF1 выражение

...

ENDIF

2.4. Считывание библиотеки макроопределений в программу

Для считывания библиотеки макроопределений в исходную программу надо передать Ассемблеру ее имя в операторе INCLUDE.

При этом оператор INCLUDE надо поместить в условную структуру IF1.

ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ.

Объектом исследования является программа на языке Ассемблер и программа-отладчик DEBUG. Работа проводится на IBM-совместимых персональных компьютерах.

ЗАДАНИЕ НА РАБОТУ

1. Внимательно ознакомиться с теоретическими положениями, ответить на контрольные вопросы.
2. Написать тексты макроопределений.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Выбрать необходимые действия в соответствии с вариантом:
 - ввод строки текста;
 - выполнение арифметических действий: сложение, вычитание, умножение, деление;
 - перевод введенных символов в число;
 - перевод полученного числового значения в символы;
 - вывод результата на экран.

	Варианты
1.	Вычислить длину окружности, если известен радиус
2.	Вычислить разность двух чисел
3.	Вычислить периметр прямоугольника, если известны длины сторон
4.	Вычислить площадь квадрата
5.	Вычислить дискриминант квадратного уравнения
6.	Вычислить длину окружности, если известен диаметр
7.	Вычислить площадь окружности, если известен радиус
8.	Вычислить площадь окружности, если известен диаметр

2. Создать библиотеку макроопределений, выполняющих выбранные действия:

3. Написать программу на Ассемблере, близкую по синтаксису к языку высокого уровня, для вычисления в соответствии с вариантом. Например, сложение трех чисел можно организовать следующей последовательностью строк:

```
read a
read b
sum a, b
read b
sum a, b
write a
```

(при условии, что результат будет храниться в переменной a). Следовательно, в библиотеке должны присутствовать макроопределения read, sum и write, выполняющие соответствующие действия. **Не забудьте**, что вводить с клавиатуры можно только символы, после чего их необходимо преобразовать в числа с помощью таблицы ASCII кодов. Обратное действие нужно проделать при выводе.

ВНИМАНИЕ! Вводить однозначные числа.

3. Оформить отчет.

ОФОРМЛЕНИЕ ОТЧЕТА

Отчет должен включать:

1. Тексты макроопределений.
2. Тексты программ.
3. Листинги трансляций.
4. Результаты работы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое макрокоманда, макрогенерация, макрорасширение, макроподстановка, макроопределение?
2. Из каких частей состоит макроопределение?
3. Каковы преимущества и недостатки использования макроопределений?
4. Какие псевдооператоры Макроассемблера вы знаете?
5. Как создать библиотеку макроопределений?
6. Как использовать библиотеку макроопределений?

2.3. Создание командных файлов

Цель и задачи работы: Приобретение практических навыков создания командных файлов.

ЗАДАНИЕ НА РАБОТУ

1. Найти в Интернете и изучить создание командных файлов под Windows и Linux
2. Составить командный файл позволяющий выполнить задание:

	Задание
1	Вывести на экран содержимое файлов, указанных в командной строке. После вывода каждого файла - комментарий и задержка до нажатия любой клавиши
2	Запустить существующий командный файл, который открывает текстовый файл. Если текстовый файл отсутствует - создать его
3	Вызвать текстовый редактор и передать ему в качестве параметра имя документа. Если при вызове не указано имя документа выводить сообщение
4	Скопировать, указанные в командной строке файлы в подкаталог с проверкой правильности записи и удалить исходные файлы. После завершения копирования каждого файла, но перед его удалением выводить предупреждающее сообщение
5	Произвести замену всех файлов на новые с проверкой всех кодов завершения операции и выдачи соответствующих комментариев
6	Проверить существуют ли файлы с таким же именем в выбранных каталогах. Если файлы существуют, вывести комментарий. Исходный - текущий каталог
7	Произвести копирование группы файлов, вводимых в командной строке при вызове пакетного файла из одного каталога в другой с удалением файлов в источнике и с выдачей предупреждающих сообщений
8	Записать каталог файлов указанного диска в файл. Шаблон файлов и дисковод задается при вызове командного файла

ВНИМАНИЕ!

Командные файлы создать для двух операционных систем: Windows и Linux.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

3. Написать, ввести и отладить командный файл в соответствии с заданием.
4. Оформить отчет.

ОФОРМЛЕНИЕ ОТЧЕТА

Отчет должен содержать название лабораторной работы, задание на работу, текст командного файла и сообщения, поступающие на экран во время работы пакетного файла.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего нужны пакетные файлы?

2. Каково основное назначение формальных параметров в пакетном файле?
3. Расшифруйте более подробно синтаксис одной из команд по указанию преподавателя.
4. Какие системные подсказки можно получить, используя команду PROMPT?
5. Каким образом число формальных параметров, используемых в пакетном файле можно увеличить до 15?
6. По каким правилам формируются метки в пакетных файлах?
7. Для чего в командных файлах используется символ @?
8. Каким образом прервать выполнение пакетного файла?
9. Каким образом приостановить выполнение пакетного файла?