

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 проведению практических (семинарских) занятий
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)

Тема 1. Управление доступом. Управление пользователями

Цель работы: изучение процесса аутентификации в UNIX.

Программа работы

Часть 1. Изучение базовых прав доступа

Начальные условия: Командная строка после входа в систему.

1. Просмотреть права доступа. Особый интерес представляют: первая колонка (права доступа), третья и четвертая – владелец и группа владельцев соответственно.
2. Рассмотреть исполняемый файл **/bin/bash**. Выяснить, чем определяются исполняемые файлы в UNIX.
3. Рассмотреть права директории **/tmp**. Найти дополнительный sticky-bit, определяющий права на создание и удаление файлов в директории.
4. Изменить файл с недостатком прав **/bin/bash**. Проанализировать все полученные предупреждения.
5. Задать текстовому файлу права только для чтения. Просмотреть установленные права доступа.
6. Лишить всех прав доступа к файлу для группы владельцев и остальных пользователей. Просмотреть установленные права доступа.

Часть 2. Изучение базы данных пользователей

Начальные условия: Командная строка после входа в систему.

1. Просмотреть содержимое файла **passwd** с помощью команды **cat /etc/passwd**. (В файле **/etc/passwd** хранятся данные о зарегистрированных в системе пользователях. Подробную информацию о формате этого файла можно получить на соответствующей странице руководств: **man 5 passwd**)
2. Просмотреть содержимое файла **group**: **cat /etc/group**. (Файл содержит данные о группах).

Часть 3. Добавление и удаление пользователей

Начальные условия: Загрузка с LiveCD (Для LiveCD суперпользователь не имеет пароля). Командная строка после входа в систему суперпользователя.

1. Добавить нового пользователя (используется команда **useradd**). В качестве параметров указать домашнюю директорию и командную оболочку пользователя: **useradd testuser -d /home/users/testuser -s /bin/sh**. Просмотреть результат: **cat /etc/passwd | grep testuser**.
2. С помощью команды **passwd** задать пароль для нового пользователя: **passwd testuser**.
3. Проверить правильность добавления нового пользователя. Войти в систему под именем вновь созданного пользователя.
4. Удалить пользователя с помощью команды: **userdel testuser**.

Отчет должен включать:

1. Название темы
2. Цель и задачи
3. Последовательность команд и результат их выполнения
4. Пояснения к результатам

Тема 2. Обработка прерываний

1. *Цель и задачи работы:* Целью и задачей работы является приобретение навыков разработки программ, использующих в своей работе функциональные клавиши для управления процессом решения.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1. Системная обработка прерываний от клавиатуры

Работая на компьютере пользователю постоянно приходится вводить с клавиатуры команды и данные. Процесс взаимодействия системы с клавиатурой представлен на рис. 1.

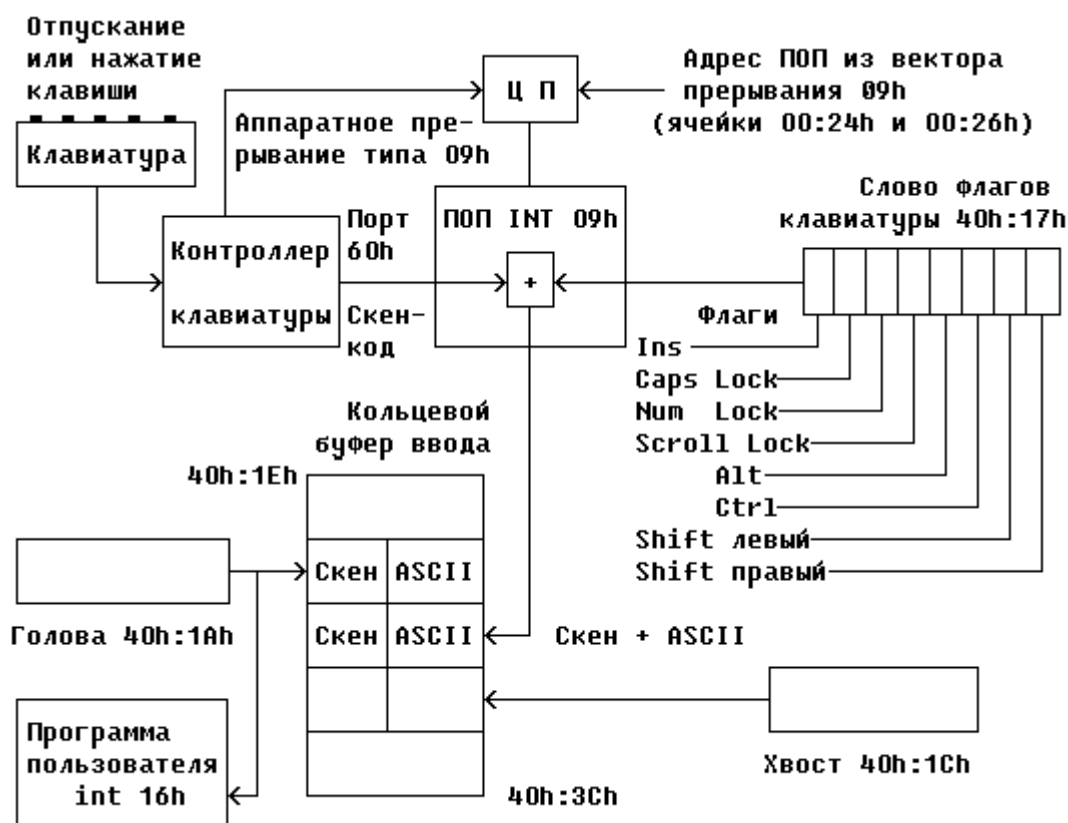


Рис. 1. Процесс взаимодействия системы с клавиатурой.

Работой клавиатуры управляет специальная электронная схема - контроллер клавиатуры. В его функции входит распознавание нажатой клавиши и помещение закрепленного за ней кода в свой выходной регистр (порт) с номером 60h. Код клавиши, поступающей в порт, называется скен-кодом и является порядковым номером клавиши. Каждой клавише присвоено как бы два скен-кода, отличающиеся друг от друга на 80h. Один скен-код (меньший, код нажатия) засылается контроллером в порт 60h при нажатии клавиши, другой (большой, код отпущения) - при ее отпущении.

Скен-код однозначно указывает на нажатую клавишу, однако по нему нельзя определить, работает ли пользователь на верхнем или нижнем регистре, а также вводит русские или латинские буквы. Очевидно, что определение введенного символа должно включать в себя не только считывание скен-кода нажатой клавиши, но и выяснение того, не были ли перед этим нажаты, например,

Shift или Caps Lock. Всем этим занимается программа обработки прерываний (ПОП) от клавиатуры.

Нажатие или отпускание любой клавиши вызывает сигнал аппаратного (внешнего) прерывания, заставляющий процессор прервать выполняемую программу и перейти на программу обработки прерывания от клавиатуры. Эта программа хранится по фиксированному адресу в постоянном запоминающем устройстве BIOS.

Процессор вместе с сигналом прерывания получает еще и тип прерывания или его номер. За клавиатурой закреплен номер 09h. Адрес программы обработки прерываний от клавиатуры располагается в векторе 09h, занимающем слова с адресами 24h и 26h.

Получив тип прерывания и определив по нему адрес вектора, процессор извлекает из вектора адрес программы обработки прерываний и осуществляет переход на ее выполнение. Программа обработки прерываний от клавиатуры вызывается через вектор 09h: INT 09h.

Программа INT 09h, помимо порта 60h, работает еще с двумя областями оперативной памяти: кольцевым буфером ввода, располагаемым по адресам от 40h:IEh до 40h:3Dh, куда помещаются коды ASCII нажатых клавиш, и словом состояния (словом флагов) клавиатуры, находящимся по адресу 40h:17h, где фиксируется состояние управляющих клавиш (Shift, Caps Lock, Num Lock и др.).

Программа INT 09h считывает из порта 60h скен-код и анализирует его значение. Если скен-код принадлежит одной из управляющих клавиш, и представляет собой код нажатия, то в слове флагов клавиатуры устанавливается бит (флаг), соответствующий нажатой клавише (биты сохраняют свое состояние пока клавиши остаются нажатыми). Например, при нажатии правой клавиши Shift в слове флагов устанавливается бит 0, при нажатии левой клавиши Shift - бит 1, при нажатии любой клавиши Ctrl - бит 2, а при нажатии Alt (тоже любой) - бит 3. Биты флагов сохраняют свое состояние пока клавиши (по одиночке или в любых комбинациях) остаются нажатыми. Если управляющая клавиша отпускается, программа INT 09h получает скен-код отпускания и сбрасывает соответствующий бит в слове флагов.

Кроме состояния указанных клавиш, в слове флагов фиксируются еще режимы Scroll Lock, Num Lock, Caps Lock и Insert, а в 101-клавишной клавиатуре на компьютерах PC/AT также состояния клавиш SysRq, Ctrl-левая, Alt-левая и режим паузы Ctrl/Num Lock.

При нажатии любой другой клавиши программа INT 09h считывает из порта 60h ее скен-код нажатия и по таблице трансляции скен-кодов в ASCII формирует двухбайтовый код (старший байт - скен-код, младший - код ASCII). При этом если скен-код характеризует клавишу, то код ASCII определяет закрепленный за ней символ. Поскольку за каждой клавишей закреплено, не менее двух символов ("a" и "A", "1" и "!", "2" и "@" и т.д.), то каждому скен-коду соответствуют, как минимум, два кода ASCII. В процессе трансляции программа INT 09h анализирует состояние флагов, так что если нажата, например, клавиша Q (скен-код 10h, код ASCII буквы Q - 51h, а буквы q - 71h), то формируется двухбайтовый код 1071h, но если клавиша Q нажата при нажатой клавише Shift, то результат

трансляции составит 1051h. Тот же код 1051B получится, если при нажатии клавиши Q был включен режим Caps Lock, однако при включенном режиме Caps Lock и нажатой клавише Shift образуется код 1071h.

Полученный в результате трансляции двухбайтовый код засылается программой INT 09h в кольцевой буфер ввода, который служит для синхронизации процессов ввода данных с клавиатуры и приема их выполняемой компьютером программой. Объем кольцевого буфера – 15 слов, причем коды символов извлекаются из него в том же порядке, в котором поступали. За состоянием буфера следят два указателя. В хвостовом указателе (слово по адресу 40:1Ch) хранится адрес первой свободной ячейки, в головном (40:1Ah) - адрес самого старого кода, принятого с клавиатуры и еще не востребованного программой. В начале работы, когда буфер пуст, оба указателя - и хвостовой, и головной, указывают на первую ячейку буфера.

Программа INT 09h, сформировав двухбайтовый код, помещает его в буфер по адресу, находящемуся в хвостовом указателе, после этого этот адрес увеличивается на 2, указывая опять на первую свободную ячейку. Каждое последующее нажатие на какую-либо клавишу добавляет в буфер очередной двухбайтовый код и смещает хвостовой указатель.

Выполняемая программа, желая получить код нажатой клавиши, должна вызвать прерывание INT 16h, которое активизирует драйвер клавиатуры BIOS. Драйвер считывает из кольцевого буфера содержимое ячейки, адрес которой находится в головном указателе, и увеличивает этот адрес на 2. Таким образом, программный запрос на ввод с клавиатуры фактически выполняет прием кода не с клавиатуры, а из кольцевого буфера.

Хвостовой указатель, перемещаясь по буферу в процессе занесения в него кодов, доходит до конца буфера (адрес 40h:3Ch). В этом случае при поступлении очередного кода адрес в указателе уменьшается на длину буфера. Тем самым указатель возвращается в начало буфера, после чего продолжает перемещаться по буферу до его конца, опять возвращается в начало и так далее по кольцу. Аналогичные манипуляции выполняются и с головным указателем.

Равенство адресов в обоих указателях свидетельствует о том, что буфер пуст. Если при этом программа вызвала прерывание INT 16h, то драйвер клавиатуры будет ждать поступления кода в буфер, после чего он будет передан в программу. Если же хвостовой указатель, перемещаясь по буферу в процессе его заполнения, подошел к головному указателю "с обратной стороны" (это произойдет, если оператор нажимает на клавиши, а программа не выполняет запросы к драйверу клавиатуры), прием новых кодов блокируется, а нажатие на клавиши возбуждает предупреждающие звуковые сигналы.

Если компьютер находится в пассивном состоянии ожидания команд DOS с клавиатуры, то за состоянием кольцевого буфера ввода следит командный процессор COMMAND.COM. Как только в буфере появляется код символа, командный процессор с помощью соответствующих системных программ переносит его в свой внутренний буфер командной строки, очищая при этом кольцевой буфер ввода, а также выводит символ на экран, организуя режим эхо-контроля. При получении кода клавиши Enter (0Dh) командный процессор

предполагает, что ввод команды закончен, анализирует содержимое своего буфера и приступает к выполнению введенной команды. При этом командный процессор работает практически лишь с младшими половинами двухбайтовых кодов символов, именно, с кодами ASCII.

Если компьютер выполняет какую-либо программу, ведущую диалог с оператором, то ввод данных с клавиатуры и вывод их на экран с целью эхо-контроля организует эта программа, обращаясь непосредственно к драйверу BIOS (INT 16h) или к соответствующей функции DOS (INT 21h). Может случиться, что выполняемой программе не требуется ввод с клавиатуры, а оператор нажал какие-то клавиши. В этом случае вводимые символы накапливаются (с помощью программы INT 09h) в кольцевом буфере ввода и не отображаются на экране. Так можно ввести до 15 символов. Когда программа завершится, управление будет передано COMMAND.COM, который сразу же обнаружит наличие символов в кольцевом буфере, извлечет их оттуда и отобразит на экране. Такой ввод с клавиатуры называют вводом с упреждением.

Имеется ряд клавиш, которым не назначены какие-то отображаемые на экране символы. Это, например, функциональные клавиши F1, F2, ..., F10; клавиши управления курсором Home, End, PgUp, PgDn, Стрелка вправо, Стрелка вниз и др. Всем этим клавишам назначены определенные скен-коды. В таблице трансляции, с которой работает программа INT 09h, всем таким скен-кодам соответствует нулевой код ASCII. Поэтому при нажатии, например, клавиши F1 (скен-код 3Bh) в кольцевой буфер ввода поступает двухбайтовый код 3B00h, а при нажатии клавиши Home (скен-код 47h) - двухбайтовый код 4700h. Двухбайтовые коды, содержащие на месте кода ASCII ноль, называются расширенными кодами ASCII. Эти коды (и соответствующие им клавиши) широко используются для управления программами.

Программы, работающие с расширенными кодами ASCII, должны, считав из кольцевого буфера ввода младший байт и убедившись, что он равен нулю, считать далее и старший байт и в зависимости от его значения выполнить соответствующее действие.

Широкое использование в компьютерах интерактивных средств потребовало расширения возможностей ввода с клавиатуры управляющей информации, которую программа должна легко отличать от вводимого текста. С этой целью в компьютерах типа IBM PC расширенные коды ASCII генерируются не только функциональными клавишами и клавишами управления курсором, но и всеми алфавитно-цифровыми клавишами, если они нажимаются вместе с клавишей Alt. Таким образом, если нажатие клавиши Q посылает в кольцевой буфер двухбайтовый код 1071h (или 1051h на верхнем регистре), то нажатие сочетания Alt/Q генерирует расширенный код ASCII 1000h, где 10 - скен-код клавиши, а 00 - признак расширенного кода ASCII. Расширенные коды ASCII генерируются также при нажатии клавиш Alt, Ctrl или Shift одновременно с функциональными клавишами F1, ..., F10. В этом случае в старший байт расширенного кода ASCII помещается уже не скен-код клавиши, а некоторый код, специально назначенный этой комбинации клавиш. Естественно, этого кода нет

среди "обычных" скен-кодов. Например, клавиша F1, скен-код которой равен 3Bh, может генерировать следующие расширенные коды ASCII:

F1	3B00h
Shift/F1	5400h
Ctrl/F1	5E00h
Alt/F1	6800h

2.2. Системные средства ввода данных с клавиатуры

DOS предоставляет три уровня процедур ввода данных с клавиатуры:

- обращение к клавиатуре, как к файлу, с помощью прерывания DOS INT 21h с функцией 3Fh;
- использование группы функций, обеспечивающих посимвольный ввод с клавиатуры в разных режимах;
- посимвольный или покодový ввод путем обращения непосредственно к драйверу BIOS с помощью прерывания INT 16h.

2.2.1. Функции DOS прерывания INT 21h.

Прерывание типа 21 (вызов функции) предоставляет множество удобных возможностей взаимодействия с клавиатурой, дисплеем, принтером, диском и асинхронным последовательным устройством. Некоторые функции, вызываемые с помощью прерывания типа 21, перечислены в табл. 1.

Таблица 1 - Вызовы функций, инициируемые прерыванием типа 21.

Регистр АН	Операция	Дополнительные входные регистры	Выходные регистры
1	Ожидает набора символа на клавиатуре и изображает его на экране (с проверкой на Ctrl-Break)	Не используют	(AL)=символ
2	Изображает на экране символ (с проверкой на Ctrl-Break)	(DL)=символ	Не использ.
5	Печать символа	(DL)=символ	Не использ.
6	Изображение символа (без проверки на Ctrl-Break)	(DL)=символ	Не использ.
6	Чтение символа с клавиатуры, (без проверки на Ctrl-Break)	(DL)=0FFH	(AL)=символ если буфер клавиатуры не пуст; (AL)=0, иначе
7	Ожидание набора символа на клавиатуре без его изображения (без проверки на Ctrl-Break)	Не используют	(AL)=символ
8	То же, что функция 7, но с проверкой на Ctrl-Break	Не используют	(AL)=символ
9	Изображает строку	(DS:DX)=адрес	Не использ.

Регистр АН	Операция	Дополнительные входные регистры	Выходные регистры
		строки, которая должна заканчиваться знаком \$	
А	Читает строку с клавиатуры в буфер	(DS:DX)=адрес буфера. Первый байт буфера = размер буфера	Второй байт буфера=число фактически прочитанных символов
В	Чтение состояния клавиатуры	Не используют	AL=0FFH,есл и клавиатурная строка пуста. AL=0, если есть хотя бы один символ
С	Опустошение буфера клавиатуры и вызов функции для работы с клавиатурой	(AL)=номер функции	В соответствии с вызываемой функцией

Для завершения программы в операционной системе DOS используются типы прерываний 10, 20 и 27. Прерывание типа 27 завершает программу с сохранением ее в памяти.

Функция 0Bh позволяет проверить наличие в кольцевом буфере ввода ожидающих символов. При обнаружении символов программа должна извлечь их из буфера одной из функций ввода; если символов нет, программа может продолжить выполнение. Такая методика используется в программах, носящих циклический характер, если требуется обеспечить управление ходом выполнения программы с клавиатуры терминала. В каждом шаге цикла после выполнения запланированных действий проверяется состояние кольцевого буфера ввода; если в течение предыдущего шага цикла оператор нажал на какую-либо клавишу, программа проанализирует введенный код и осуществит выход из цикла и переход в ту или иную точку; если же буфер оказывается пуст, циклическое выполнение продолжится.

Функция 0Bh чувствительна к Ctrl/C. Это дает возможность организовать с ее помощью аварийное завершение программы на тех ее участках, где выполняются чисто процессорные действия. Если, например, включить вызов функции 0Bh в цикл, то при отсутствии ввода с клавиатуры цикл будет выполняться обычным образом, но после ввода Ctrl/C программа аварийно завершится, хотя на выполняемом участке программы не используются функции ввода-вывода.

Функция 0Ch служит для организации ввода с предварительной очисткой кольцевого буфера.

Все функции, кроме 0Ch, вводят в программу наиболее старый из скопившихся в кольцевом буфере ввода символов, реализуя тем самым возможность ввода с упреждением. В этом режиме оператор может нажимать на клавиши еще до выдачи программой запроса на ввод; коды нажатых клавиш (не более 15) будут накапливаться в кольцевом буфере ввода и извлекаться оттуда в программу по мере выполнения ею запросов на ввод. В отличие от этого, функция 0Ch сначала очищает кольцевой буфер и лишь затем ожидает ввода символа с клавиатуры. В результате коды всех ранее нажатых (по предположению - случайно) клавиш теряются. Обычно функция ввода 0Ch стоит в программе непосредственно вслед за функцией вывода на экран символьной строки с предложением оператору вводить данные. В результате из кольцевого буфера убирается весь "мусор" от случайных нажатий, в программу же поступает лишь то, что вводится оператором после запроса программы. При этом режим ввода (с эхом или без него и т.д.) определяется тем, какая именно функция ввода (01h, 06h, 07h, 08h или 0Ah) реализуется "внутри" функции 0Ch.

Сравнительные характеристики функций DOS ввода с клавиатуры приведены в табл. 2.

Таблица 2

	01h	06h	07h	08h	0Ah	0Bh	0Ch
Эхо	+	+	-	-	+	+/-	
Реакция на ^C	+	-	-	+	+	+	+/-
Перенаправление	+	+	+	+	+	+	+
Ожидание символа	+	-	+	+	+	-	+/-
Расширенные ASCII	+	+	+	+	-		+
Alt+код	-	+	+	+	-		+/-
Очистка буфера	-	-	-	-	-		+

Все функции DOS ввода с клавиатуры допускают перенаправление ввода (из файла, последовательного порта, из вывода другой программы). Если требуется избавиться от этого качества, следует использовать файловую функцию ввода 3Fh и специально выделенный дескриптор.

2.2.2. Ввод с клавиатуры средствами файловой системы

Ввод с клавиатуры средствами файловой системы (INT 21h, функция 3Fh) осуществляется точно так же, как и чтение из файла. Обычно используется предопределенный дескриптор 0, закрепленный за стандартным устройством ввода (по умолчанию за клавиатурой). Число вводимых символов указывается в регистре CX, однако ввод завершается лишь после того, как нажата клавиша Enter, независимо от того, введено ли фактически меньше символов, чем было запланировано, или больше (при неправильных действиях). Поэтому при вводе строк с клавиатуры нет необходимости заранее задавать их длину, достаточно загрузить в регистр CX максимальную длину строки, например, 80 байт. В любом случае в регистре AX возвращается число реально введенных байтов, при этом

учитываются также и два байта (0Ah и 0Dh), поступающие во входной буфер при нажатии клавиши Enter.

Особая ситуация возникает, если попытаться ввести больше символов, чем затребовано функцией 3Fh. В процессе выполнения этой функции все вводимые символы тут же извлекаются из кольцевого буфера ввода и пересылают в буфер DOS. Обнаружив во входном потоке коды клавиши Enter, DOS пересылает из этого буфера в буфер пользователя в программе точно затребованное число символов (естественно, без кодов Enter, которые располагаются в конце вводимой строки). Остальные символы остаются в буфере DOS, готовые к вводу. Фактически, если не принять специальных мер к очистке буфера, они поступят в программу при очередном запросе 3Fh, даже если оператор еще не начал вводить очередную порцию данных. Очевидно, что в этом случае будет нарушена синхронизация кода выполнения программы с работой оператора.

Поскольку дескриптор 0 закреплен не собственно за клавиатурой, а за стандартным устройством ввода, он обеспечивает перенаправление ввода. Пусть, например, программа PROCDAT содержит строки ввода через дескриптор 0. При запуске программы командой

PROC DAT

она будет ожидать ввода исходных данных с клавиатуры, однако запуск ее командой

PROC DAT<RAW DAT.001

приведет к автоматическому вводу информации из файла RAW DAT.001, который в данном случае ищется в текущем каталоге текущего диска. Поиск и открытие файла осуществляет DOS.

Иногда требуется выключить механизм перенаправления (возможно, лишь для определенных операторов ввода). Для этого следует открыть консоль для ввода как файл (с именем CON) функцией 3Dh, получить выделенный системой дескриптор, а затем использовать его в операциях ввода 3Fh:

;Поля данных			
keybd	db	'CON',0	;Имя устройства
handle	dw	0	;Новый дескриптор
;Откроем новый дескриптор			
	mov	AH,3Dh	;Функция открытия
	mov	AL,0	;Доступ для чтения
	mov	DX,offset keybd	;Адрес имени устройства
	Int	21h	
	Mov	handle,AX	;Получили дескриптор

2.2.3. Работа с клавиатурой на уровне BIOS

Работа с клавиатурой на уровне BIOS (INT 16h) позволяет считывать двухбайтовые коды, поступающие в кольцевой буфер ввода (код ASCII + скен-код) и анализировать слово флагов клавиатуры (нажатие клавиш Shift, Caps Lock и др.). Для ввода используются следующие функции прерывания INT 16h:

00h - чтение двухбайтового кода из входного буфера;

01h - чтение состояния клавиатуры и двухбайтового кода без извлечения его из буфера;

02h - чтение флагов клавиатуры,

Функция 00h позволяет в одном действии получить полный двухбайтовый код нажатой клавиши и комбинации клавиш, из которого, в частности, можно извлечь скен-код (некоторые программы идентифицируют нажатые клавиш не по кодам ASCII, а по их скен-кодам), а также получить значащую часть расширенного кода ASCII (при нажатии, например, функциональных клавиш). Функция 00h является синхронной: при ее выполнении программа останавливается в ожидании нажатия клавиши.

Функция 01h относится к числу асинхронных: определив состояние клавиатуры (точнее - буфера ввода), она возвращает управление программе. Состояние буфера возвращается в флаге ZF: если в буфере имеются ожидающие ввода в программу символы, ZF=0, если же буфер пуст, ZF=1. При наличии в буфере кода символа его можно проанализировать, так как он возвращается функцией в регистре AX (AH=скен-код, AL=код ASCII). Необходимо однако иметь в виду, что функция 01h, копируя двухбайтовый код в регистр AX, не очищает при этом кольцевой буфер. Забрать символ с очисткой буфера можно затем функцией 00h.

Функция 02h - чтение флагов клавиатуры - передает в программу содержимое слова флагов (ячейка 417h). Она может использоваться программами, работающими на уровне скен-кодов, для определения состояния клавиш Shift, Caps Lock и др.

2.3. Примеры программирования ввода с клавиатуры

Пример 1. Чтение символа с клавиатуры.

```
GET_KEY MOV     AH, 1      ; считать символ
          INT     21H
          CMP     AL, 'Д'   ; он равен "Д"?
          JE      YES      ; если да, перейти на метку YES
          CMP     AL, 'Н'   ; он равен "Н"?
          JE      NO       ; если да, перейти на метку NO
          JNE     GET_KEY  ; иначе ждать ввода "Д" или "Н"
```

Пример 2. Чтение строки с клавиатуры.

```
U_STR    DB 50,51 DUP(?) ; резервируется блок байтов, длина которого на 2
                                     ; байта превышает максимальное число символов в
                                     ; строке. Первый байт задает длину строки (50)
```

...

```
          PUSH    AX
          LEA     DX, U_STR    ; считать строку
          MOV     AH, 0AH
          INT     21H
          SUB     CH, CH
          MOV     CL, U_STR+1  ; счетчик символов в CX
```

ADD	DX,2	; DX-указатель на начало строки
POP	AX	

Пример 3. Ввод с клавиатуры и вывод на экран символьной информации. Вывести на экран через дескриптор стандартной ошибки служебное сообщение. Ввести с клавиатуры через дескриптор стандартного ввода символьную строку, вывести ее на экран через дескриптор стандартного вывода.

;Определения

STDIN	EQU 0	;дескриптор стандартного ввода
STDOUT	EQU 1	;дескриптор стандартного вывода
STDERR	EQU 2	;дескриптор стандартной ошибки

;Основные фрагменты программы

;Выведем служебное сообщение msg

...

;Поставим запрос на ввод строки в буфер buf

MOV AH,3Fh	;Функция ввода
MOV BX,stdin	;дескриптор стандартного ввода
MOV CX,80	;Ввод максимум 80 байт
MOV DX, offset buf	;Адрес буфера ввода
INT 21h	
MOV actlen, AX	;Фактически введено

;Выведем введенное на экран

MOV AH,40h	;Функция вывода
MOV BX, stdout	;дескриптор стандартного вывода
MOV CX,actlen	;Длина сообщения
MOV DX, offset buf	;Адрес сообщения
INT 21h	

;Завершим программу

...

;Поля данных

msg	db	'Вводите! '	
msglen	equ	\$-msg	;Длина сообщения
buf	db	80 dup (0)	;Буфер ввода
actlen	dw	0	

Пример 4. Ввод с клавиатуры, обработка и вывод на экран символьной информации. Усложнить программу из примера 1, предусмотрев фильтрацию входного символьного потока, например, преобразование строчных латинских (или русских, или и тех, и других) букв в прописные.

; Определения stdin, stdout, stderr

...

; Основные фрагменты программы

; Выведем служебное сообщение msg

...

; Поставим запрос на ввод строки в буфер buf, отправив длину

```

; фактически введенной строки actlen
...
;Превратим строчные латинские буквы в прописные
      MOV CX, actlen      ;Длина введенной строки
      MOV SI,0            ;Указатель в буфере
filter: MOV AL, buf[SI]    ;Возьмем символ
      CMP AL, 'a'         ;Меньше 'a'?
      JB  nolet           ;Да, не преобразовывать
      CMP AL, 'z'         ;Больше 'z'?
      JA  nolet           ;Да, не преобразовывать
      SUB AL,20h          ;Преобразуем в прописную
      MOV buf[SI], AL     ;И отправим назад в buf
nolet: INC  SI            ;Сместим указатель
      LOOP filter         ;Цикл
;Выведем введенное на экран
...
;Завершим программу
...
;Поля данных
msg      db  'Вводите! '
msglen   equ  $-msg      ;Длина сообщения
buf      db  80 dup(' ') ;Буфер ввода
actlen   dw  0

```

Пример 5. Управление программой от функциональных клавиш. С помощью функции 08h организовать ввод в программу управляющих кодов от функциональных клавиш F1, ..., F10 или других клавиш, дающих расширенные коды ASCII (сочетания Alt/буква, Alt/цифра и др.). Вводимые коды использовать для управления ходом программы.

;Основные фрагменты программы

;Ожидаем нажатия клавиши

```
again:  MOV AH,08h
```

;Функция ввода без эха

```
      INT 21h
```

```
      CMP AL,0      ;Младший байт кода = 0?
```

```
      JNE again     ;Нет, повторить
```

```
      MOV AH, 08h   ;Да, введем старший байт кода
```

```
      INT 21h
```

```
      CMP AL,59     ;Нажата F1?
```

```
      JE  f1        ;Да
```

```
      CMP AL,84     ;Нажаты Shift/F1?
```

```
      JE  shiftf1    ;Да
```

```
      CMP AL,30     ;Нажаты Alt/A?
```

```
      JE  alta      ;Да
```

```
      CMP AL,120    ;Нажаты Alt/I?
```

```

                JE    altl          ;Да
                JMP   again         ;Нажато незапланированное
; Вывод соответствующих сообщений
fl:
; Вывод сообщения mfl
...
                JMP   outpr
shiftfl:
;Вывод сообщения mshiftfl
...
                JMP   outpr
alta:
; Вывод сообщения malta
...
                JMP   outpr
altl:
; Вывод сообщения maltl
...
                JMP   outpr
outpr:
; Завершим программу
...
;Поля данных
mfl            db    'Введено F1'
mflen         equ    $-mfl
mshiftfl       db    'Введено Shift/F1'
mshflen       equ    $-mshiftfl
malta          db    'Введено Alt/A'
maltalen      equ    $-malta
maltl          db    'Введено Alt/I'
maltlen       equ    $-maltl

```

Пример 6. Ввод с клавиатуры с предварительной очисткой буфера. Организовать цикл ввода в программу данных по ее запросу. Анализировать введенный символ. Если введено "Q", завершить программу.

```

;Основные фрагменты программы
;Выведем запрос req на ввод символа
...
;Очистим буфер ввода и поставим запрос на ввод с клавиатуры
again:    MOV AH,0Ch          ;Функция ввода с очисткой буфера
          MOV AL,01h          ;Выберем функции ввода 01h
          INT  21h             ;(можно 01h, 06h,07h,08h)
;Проанализируем введенное
          CMP AI,'Q'           ;Введено "Q"?
          JE    outprog        ;Да

```

```

        JMP again
outprog:
;Завершим программу
...
;Поля данных
req      db    10,13, 'Вводите команду:'
reqlen   equ   $-req

```

Пример 7. Чтение двухбайтового кода из кольцевого буфера ввода.

```

;Основные фрагменты программы
;Будем ждать ввода символа
again:   MOV AH,00h      ;Функция чтения двухбайтового кода
        INT 16h
        JMP again

```

Пример 8. Управление циклической программой от клавиатуры с помощью функции чтения состояния клавиатуры. Организовать выход из циклического участка программы при нажатии на любую клавишу.

```

;Определение
stdout   equ 1           ;Дескриптор стандартного вывода
;Основные фрагменты программы
;Организуем цикл каких-нибудь действий с периодическим опросом клавиатуры
;Конкретные действия - вывод символа на экран с небольшой задержкой
again:   MOV AH, 40h      ;Функция вывода
        MOV BX, stdout    ;Дескриптор стандартного вывода
        MOV CX, 1         ;Один символ
        MOV DX, offset sym ;Адрес символа
        INT 21h
;Организуем небольшую задержку
...
;Получим состояние клавиатуры
        MOV AH,01h       ;Функция чтения состояния
        INT 16h          ;клавиатуры
        JZ again         ;Если Z=1, символа нет
;Сообщим о выходе из цикла с помощью функции 40h
...
;Заберем введенный символ из кольцевого буфера ввода
        MOV AH,00h       ;Получим символ в AX
        INT 16h          ;и пусть он пропадет
;Завершим программу
...
;Поля данных
sym      db    '*'
mes      db    'Программа завершена оператором'

```

Пример 9. Изображение строки на экране.

PROMPT DB 'Введите вашу фамилию : \$'

...

```
LEA    DX, PROMPT
MOV     AH,9
INT     21H
```

Примечание: 1. Курсор останется в позиции символа \$;
2. Команда LEA (load effective address - загрузить исполнительный адрес) пересылает смещение ячейки памяти в любой 16-битовый регистр общего назначения, регистр указателя или адресный регистр.

3. Чтобы после выдачи строки символ курсор перешел на начало следующей строки экрана, надо перед знаком \$ вставить символы возврата каретки (0DH) и перехода на следующую строку (0AH), например:

MES DB 'Операция завершена',0DH,0AH,'\$'

2.4. Особенности создания программы с расширением COM

В отличие от программы типа .EXE программа типа .COM содержит лишь один сегмент, включающий все компоненты программы. Структура программы типа .COM выглядит следующим образом:

```
Title    COM-программа
Text     segment    'code'
          Assume     cs:text, ds:text, es:text, ss:text
          Org        100h
Myproc   proc
...       ;Текст программы
myproc   endp
...       ;Данные
text     ends
end
```

Трансляция и компоновка такой программы проходит обычным образом, после чего ее надо преобразовать в файл типа .COM командой EXE2BIN. Например, чтобы преобразовать файл p.exe, надо выполнить команду

EXE2BIN p p.com

2.5. Видеосистема компьютеров типа IBM PC

Видеосистема компьютера включает в себя ряд аппаратных и программных средств, позволяющих получать на экране терминала текстовые и графические изображения.

К аппаратным средствам можно отнести сам видеотерминал (монитор, дисплей) - высококачественную электронно-лучевую трубку со схемами питания и управления, а также видеоконтроллер, или видеоадаптер - электронную плату, обеспечивающую вывод на экран текстовых и графических изображений, а также программное управление видеосистемой. Качество и возможности изображения в значительной степени определяются характеристиками видеоадаптера.

В настоящее время широко используется адаптер SVGA. Адаптер обеспечивает два программно переключаемых режима - текстовый и графический.

В графическом режиме на экран выводится цветное изображение, причем в памяти адаптера могут одновременно храниться несколько независимых изображений (графических страниц).

В текстовом режиме изображение обычно состоит из 25 строк по 80 символов в каждой, хотя имеется возможность увеличить число строк до 40 за счет уменьшения высоты отображаемых символов. Каждый символ и фон под ним могут принимать любой из 256 цветов.

Поскольку таблицы, описывающие форму символов, загружаются в память адаптера программно, имеется возможность работать с символами любой конфигурации (например, греческие буквы). Обычно используется стандартная кодовая таблица символов, содержащая знаки английского и русского алфавитов, цифры, знаки препинания, специальные машинные знаки, символы псевдографики для рисования в текстовом режиме разнообразных рамок, а также некоторые математические символы.

В памяти адаптера одновременно может храниться до 8 текстовых страниц. Переключение как текстовых, так и графических страниц осуществляется программно.

Программные средства обслуживания экрана включают в себя

- **видеодрайвер BIOS**, к которому можно обратиться из прикладной программы с помощью прерывания INT 10h, и который обеспечивает нижний уровень управления (вывод символов, работа с курсором, переключение режимов видеоадаптера и т.д.),
- **программы DOS**, активизируемые с помощью прерывания INT 21h и предоставляющие более высокий уровень сервиса в текстовом режиме.

Работа в графическом режиме не поддерживается DOS и может осуществляться только с помощью функций видеодрайвера BIOS.

2.6. Вывод на экран средствами DOS

DOS предоставляет следующие возможности вывода текстовой информации на экран:

- использование группы функций DOS INT 21h, реализующих посимвольный вывод, а также вывод строк;
- обращение к экрану как к файлу, с помощью прерывания DOS INT 21h с функцией 40h.

2.6.1. Функции 21 прерывания DOS.

При выводе информации на экран можно использовать функции 2h, 5h, 6h и 9h прерывания DOS типа 21h (см. табл. 7.2 и пример 7.9 лабораторной работы 7.1).

2.6.2. Вывод на экран средствами файловой системы

Вывод на экран средствами файловой системы (INT 21h, функция 40h) осуществляется точно так же, как и запись в файл. Используются

предопределенные дескрипторы 1 или 2, закрепленные за стандартными устройствами вывода и ошибки, соответственно (по умолчанию - экраном). Число выводимых символов указывается в регистре CX, а адрес выводимой строки - в регистре DX. Коды 08h (забой), 0Ah (перевод строки), 0Dh (возврат каретки) и некоторые другие рассматриваются как управляющие и приводят к выполнению соответствующих им действий. Число выводимых символов передается через регистр CX, однако, если в строке встречается <Ctrl>/Z (код 26), вывод прекращается. Дескриптор 1, закрепленный за стандартным устройством вывода> обеспечивает перенаправление вывода. Пусть, например, программа FILTXT содержит строки вывода через дескриптор 1. При запуске программы командой

```
FILTXT
```

ее вывод появится на экране, однако команда

```
FILTXT > MYFILE.DOC
```

приведет к автоматическому образованию файла MYFILE.DOC и записи в него всего вывода программы (на экран ничего не поступит); команда

```
FILTXT >> MYFILE.DOC
```

направит вывод программы FILTXT в тот же файл MYFILE.DOC, при этом новая запись добавится к уже существующим, увеличив тем самым длину файла MYFILE.TXT.

Предопределенный дескриптор 2 (стандартная ошибка) всегда связан с экраном и не может быть перенаправлен.

Если требуется исключить для каких-то операторов вывода или для всей программы в целом возможность перенаправления, то, помимо использования дескриптора стандартной ошибки, можно открыть экран, как файл (с именем CON) функцией 3Dh, получить выделенный системой дескриптор, а затем использовать его в операциях вывода 40h:

```
;Поля данных
screen          db    'CON', 0          ;Имя устройства
handlscr        dw    0                  ;Новый дескриптор
;Откроем новый дескриптор
                MOV AH, 3Dh              ;Функция открытия
                MOV AL, 1                  ;Доступ для записи
                MOV DX, offset screen     ;Адрес имени устройства
                INT  21h
                MOV handlscr, AX          ;Получили дескриптор
```

Средства DOS "в чистом виде" позволяют выводить на экран только черно-белый текст; возможности позиционирования текста на экране ограничиваются использованием символов возврата каретки (0Dh) и перевода строки (0Ah). Для вывода на экран цветных изображений следует использовать управляющие Esc-последовательности, реализуемые драйвером ANSI.SYS.

2.7. Управление экраном через ANSI-драйвер

Включение в систему ANSI-драйвера терминала (файл ANSI.SYS) дает пользователю дополнительные возможности управления экраном и клавиатурой.

Если в символьной строке, выводимой на экран, встречается код клавиши <Esc> (27 десятичное, или 1B шестнадцатеричное), за которым следует символ [, то ANSI-драйвер перехватывает последующие символы и интерпретирует их как команды управления экраном или клавиатурой. С помощью Esc-последовательностей можно очищать экран, перемещать по нему курсор, выбирать цвета фона и символа, изменять видеорежим, а также переопределять клавиши клавиатуры.

Группа Esc-последовательностей обеспечивает управление курсором. Так, Esc[2J очищает экран, Esc[строка;столбецN позиционирует курсор и т.д. Перечень этих последовательностей приведен ниже.

Управление цветом предполагает возможность задания трех параметров: цвета фона, цвета символа и дополнительной характеристики символа (мерцание, повышенная яркость и др.). Поэтому соответствующая Esc-последовательность имеет вид

Esc[код-1;код-2;код-3m

где код-1, код-2 и код-3 - коды трех указанных параметров в любом порядке. При необходимости можно указать только два или один параметр:

Esc[7m Символы становятся мерцающими;

Esc[34;7m Голубые мерцающие символы;

Esc[47;7;34m Голубые мерцающие символы на белом фоне.

Esc[0m Отмена заданных ранее цвета и характеристики символов, возвращая комбинацию: белые символы на черном экране.

Следует иметь в виду, что ANSI-драйвер чувствителен к регистру клавиатуры, на котором вводится завершающая Esc-последовательность буква. Так Esc-последовательность Esc[2J завершается обязательно прописной буквой J, а Esc-последовательность Esc[0m - строчной буквой m.

Esc-последовательности часто используются в обучающих, игровых и сервисных программах для формирования на экране требуемого изображения. В этом случае Esc-последовательности включаются в строки, выводимые на экран операторами вывода того языка, на котором написана программа (Паскаль, Си, Ассемблер и пр.). Это дает возможность выводить последовательные строки текста в разные места экрана, изменять их цвет, заставлять мигать или выделяться яркостью и т.д.

Для управления экраном используются такие Esc-последовательности:

Esc[2J - очистка экрана и перемещение курсора в левый верхний угол;

Esc[K - очистка строки от курсора до конца строки;

Esc[строка;позицияN - установка позиции курсора. Параметр строка обозначает Y-координату курсора в пределах 1-25, параметр позиция - X - координату в пределах 1-80 (для видеорежима 80x25 символов), Вместо символа N в конце последовательности можно использовать символ f;

Esc[числоA - перемещение курсора на число строк вверх;

Esc[числоB - перемещение курсора на число строк вниз;

Esc[числоC - перемещение курсора на число позиций вправо;

Esc[числоD - перемещение курсора на число позиций влево;
 Esc[6n - вывод текущих координат курсора в формате
 Esc[строка;позицияR;
 Esc[s - сохранение текущих координат курсора в специальном
 буфере;
 Esc[u - восстановление сохраненных в буфере координат
 курсора (используется вместе с предыдущей
 последовательностью для того, чтобы вывести что-то на
 экран в другом месте, а потом вернуть курсор в
 прежнюю позицию);

Esc[код-1;код-2;код-3т - выбор атрибутов символов. Параметры код-1, код-
 2 и код-3 могут принимать значения:

- 0 - нормальное изображение (белые символы на черном поле);
- 1 - выделение яркостью;
- 4 - выделение подчеркиванием (только для монохромных дисплеев);
- 5 - выделение мерцанием;
- 7 - инверсное изображение (черные символы на белом поле);
- 8 - скрытый текст (только для монохромных дисплеев; цветное изображение
 можно скрыть, установив одинаковые цвета фона и символа):

30 - черные символы	40 - черный фон
31 - красные символы	41 - красный фон
32 - зеленые символы	42 - зеленый фон
33 - коричневые символы	43 - коричневый фон
34 - синие символы	44 - синий фон
35 - фиолетовые символы	45 - фиолетовый фон
36 - бирюзовые символы	46 - бирюзовый фон
37 - белые символы	47 - белый фон.

Помимо перечисленных, имеются Esc-последовательности, служащие для
 выбора видеорежима, переопределения клавиш клавиатуры и другие.

Все перечисленные выше функции DOS позволяют включать в состав
 выводимой строки Esc-последовательности. Например, для размещения в центре
 пустого экрана фразы "Возможности:", написанной красным цветом по белому
 полю, надо выполнить функцию 40h прерывания 21h (вывод в файл или
 устройство) с предопределенным дескриптором 1, выводимую же строку (адрес
 которой помещается в регистр DX) сформировать следующим образом:

```
mes      db      27,'[2J',27,'[12;32H',27,'[31;47m','Возможности:',27,'[0m'
```

В строку включены 4 Esc-последовательности:

Esc[2J - очистка экрана;
 Esc[12;32H - установка курсора (12-я строка, 32-й столбец);
 Esc[31;47m - установка цвета сообщения (красные символы по белому
 полю);
 Esc[0m - отмена цвета, чтобы остальные выводимые на экран строки
 оставались черно-белыми.

2.8. Вывод на экран средствами BIOS

Рассмотрим теперь средства вывода на экран, реализуемые драйвером BIOS, программное обращение к которому осуществляется с помощью прерывания INT 10h. В таблице 3. представлены операции, инициируемые прерыванием типа 10 в черно-белом текстовом режиме с решеткой 80×25.

Таблица 3

АН	Операция	Дополнительные входные регистры	Выходные регистры
2	Перемещение курсора в заданную позицию	(DH,DL)=(строка, столбец) (0-24,0-79) (BH)=номер страницы (0-4)	Не использ.
3	Чтение положения курсора	(BH)=номер страницы (0-4) (DH,DL)= строка, столбец курсора	(CH,CL)= текущий режим курсора
5	Задание новой активной страницы дисплея	(AL)=новый номер страницы (0-3)	Не использ.
6	Прокрутка активной страницы вверх	(AL)=число строк Строки появляются внизу окна (AL)=0 - пробелами все окно. (CH,CL)=строка, столбец верхнего левого угла прокручиваемого окна (DH,DL)=строка, столбец нижнего правого угла прокручиваемого окна (BH)=атрибут, используемый при изображении строки пробелов	Не использ.
7	Прокрутка активной страницы вниз	Те же, что и выше, но строки пробелов входят в окно сверху	Не использ.
Процедуры обработки символов			
8	Чтение символа, находящегося в текущей позиции курсора и его атрибута	(BH)=номер страницы дисплея (0-3)	(AL)=считанный символ (AH)=атрибут этого символа
9	Запись символа и нового атрибута в текущую позицию курсора	(BH)=номер страницы дисплея (0-3) (BL)=атрибут символа (CX)=счетчик записываемых символов (AL)=записываемый символ	Не использ.
A	Запись символа без изменения атрибута в текущую позицию курсора	(BH)=номер страницы дисплея (0-3)- (CX)=счетчик записываемых символов (AL)=записываемый символ	Не использ.
Процедура ASCII-телетайп для вывода			

АН	Операция	Дополнительные входные регистры	Выходные регистры
Е	Вывод на экран и перемещение курсора в следующую позицию	(AL)=записываемый символ (BH)=номер страницы дисплея (0-3)	Не использ.
Чтение видеостатуса			
Ф	Чтение текущего видеостатуса	Не используется	(AL)=текущий режим (АН)=число столбцов на экране (BH)=текущая страница дисплея

Функция 02h позволяет позиционировать текстовый курсор, задавая его местоположение в виде номера строки (0...24) и номера столбца (0...79). Видеодрайвер поддерживает 8 независимых курсоров - по одному на каждую страницу, причем функция 02h позиционирует курсор независимо от того, какая страница является активной.

Функция 03h позволяет получить и сохранить текущее положение курсора. Это дает возможность перейти временно в другое место экрана, сформировать там изображение, а затем вернуться на старое место.

Функция 05h переключает видеостраницы дисплея. Если дисплей находится в текстовом режиме, то переключаются текстовые страницы (0...7), если установлен графический режим, то переключаются графические страницы (0...1).

Большая часть описываемых ниже функций вывода на экран (кроме подфункции переключения мерцания/яркости и функции вывода строки в режиме телетайпа) позволяют формировать изображение на любой видеостранице, как активной в настоящий момент, так и скрытой. Это дает возможность либо подготовить заранее несколько страниц и по мере необходимости быстро их переключать, либо, пока одна страница выводится на экран, готовить изображение на следующей.

С помощью функций 06h и 07h в заданном месте экрана дисплея создаются цветные прямоугольные окна заданного размера. Если в созданные ранее окна выведен какой-либо текст, то с помощью этих же функций можно прокручивать текст вверх или вниз. При этом текст, уходящий за край окна, пропадает, а из-под противоположного края появляются пустые строки с заданными атрибутами цвета. Для заполнения появляющихся строк текстом следует использовать подходящие функции DOS или BIOS, причем контроль местоположения, длины и цвета строк возлагается на программиста. Драйвер только прокручивает заданную прямоугольную область экрана (вместе с текстом в ней).

Функции 09h, 0Ah, 0Eh и 13h служат для вывода на экран отдельных символов и символьных строк (в цикле). Функции 09h и 0Ah не выполняют

фильтрации управляющих символов, поэтому с их помощью можно выводить все символы кодовой страницы. Предусмотрен вывод одного и того же символа заданное число раз, что можно использовать при создании рамок и других орнаментов. Вывод символа не перемещает курсор, поэтому каждый раз перед применением функций 09h или 0Ah следует позиционировать курсор с помощью функции 02h. Различие функций 09h и 0Ah заключается в том, что первая позволяет вывести символ с любым атрибутом, а вторая использует прежний атрибут той позиции, куда выводится символ.

Функция 0Eh фильтрует управляющие коды 07h (звуковой сигнал), 08h (возврат на шаг), 10h (перевод строки) и 13h (возврат каретки), выполняя соответствующие им действия. Курсор перемещается после вывода каждого символа, что дает возможность выводить целые строки. Однако атрибут символа установить нельзя, выводимый символ приобретает прежний атрибут той позиции, куда он выводится. При необходимости вывода символа с новым атрибутом следует сначала вывести в заданную позицию символ пробела с требуемым атрибутом (функцией 09h), а затем туда же послать символ с помощью функции 0Eh.

Важным свойством функции 0Eh является автоматический переход на следующую строку после завершения предыдущей, а также прокрутка экрана вверх на одну строку после заполнения самой нижней строки.

Функция 13h предназначена для вывода строк с указанием атрибутов как каждого символа в отдельности, так и всей строки. Функция может выполняться в четырех вариантах в зависимости от кода режима, указываемого в регистре AL. В режимах 0 и 1 атрибут символов указывается сразу для всей строки в регистре BL, причем в режиме 0 курсор не смещается в процессе вывода, а в режиме 1 - смещается на длину строки. В режимах 2 и 3 атрибуты символов включаются в выводимую строку, в которой, таким образом, чередуются коды атрибутов и коды символов, что усложняет формат строки, но позволяет устанавливать атрибуты для каждого символа независимо. Режим 2 отличается от режима 3 тем, что в первом случае курсор не смещается, а во втором смещается на длину строки.

При вызове функции 13h в регистре DX задаются координаты начала выводимой строки (в DH - строка экрана и в DL - столбец), а в регистре CX - длина выводимой строки, которая в режимах 2 и 3 оказывается за счет байтов с атрибутом в два раза больше длины строки, реально появляющейся на экране. Несколько необычно указывается адрес выводимой строки. Она должна быть помещена в регистры ES:BP (ES - сегментный адрес и BP - смещение в пределах сегмента).

Функция 13h выводит не все символы, так как коды 07h, 08h, 0Ah и 0Dh рассматриваются ею, как управляющие.

При выводе на экран средствами драйвера BIOS необходимо иметь в виду, что ввод с клавиатуры <Ctrl>/C не приводит к завершению программы. Следует опасаться бесконечных циклов вывода на экран - выход из них возможен только путем перезагрузки компьютера.

Подфункция 03h функции 10h (прерывание 10h), в отличие от описанных выше функций вывода символов и строк, воздействует сразу на весь экран, влияя

на отображение тех символов, у которых установлен старший бит атрибута фона. функция позволяет либо приписать этот бит яркости фона, давая тем самым возможность выводить на экран 16 цветов фона, либо назначить его атрибуту мерцания символа. В последнем случае цвет фона может принимать только 8 значений.

2.9. Примеры программирования вывода на экран

Пример 10. Вывести на экран все 256 символов кодовой таблицы четырьмя строками по 64 символа.

;Основные фрагменты программы

;Очистим экран и зададим атрибуты символов с помощью окна

MOV AH,06h	;Функция инициализации окна
MOV AL,0	;Не прокручивать
MOV BH,31h	;Бирюзовый фон, синие символы
MOV CH,0	;Y левый верхний
MOV CL,0	;X левый верхний
MOV DH,24	;Y правый нижний
MOV DL,79	;X правый нижний
INT 10h	

;Будем выводить символы четырьмя строками по 64 символа

MOV CX,4	;4 строки (внешний цикл)
----------	--------------------------

12: PUSH CX ;Сохраним счетчик внешнего цикла

MOV CX,64	;64 столбца (внутренний цикл)
-----------	-------------------------------

11: PUSH CX ;Сохраним счетчик внутреннего цикла

;1) Позиционируем курсор

MOV AH,02h	;Функция позиционирования
MOV BH,0	;Страница 0
MOV DH,row	;Строка
MOV DL,col	;Столбец
INT 10h	

;2) Выведем очередной символ

MOV AH,0Ah	;Функция вывода символа
MOV AL,char	;Код символа
MOV BH,0	;Страница 0
MOV CX,1	;Коэффициент повторения
INT 10h	

;3) Установим новые значения переменных и организуем циклы

INC char	;Следующий символ
INC col	;Следующий столбец
POP CX	;Восстановим счетчик

;внутреннего цикла

LOOP 11	
MOV col,10	;Снова с новой строки
ADD row,2	;Вниз на две строки
POP CX	;Восстановим счетчик

```

;внешнего цикла
        LOOP I2
;Завершим программу
...
;Поля данных
row      db    10      ;Начальная строка
col      db    10      ;Начальный столбец
char     db    0       ;Начальный код символа

```

Пример 11. Вывод строки в режиме телетайпа. Вывести строку символов с указанием атрибутов.

```

;Основные фрагменты программы
        MOV AX,DATA      ;Инициализируем
        MOV DS,AX        ;оба сегментных регистра:
        MOV ES,AX        ;DS и ES
;Выведем строку
        MOV AH,13h       ;Функция вывода строки
        MOV AL,3         ;Формат строки (коды/атрибуты)
        MOV BH,0         ;Страница 0
        MOV CX,numsym    ;Число символов
        MOV DH,24        ;Строка
        MOV DL,20        ;Столбец
        MOV BP, offset string ;Адрес строки
        INT 10h
;Завершим программу
...
;Поля данных
;Переключаем коды и атрибуты
string   db    16,42h,16,42h,16,42h
         db    'S',28h,'T',28h,'R',28h,'T',28h,'N',28h,'G'
         db    28h,17,42h,17,42h,17,42h
stringlen equ    $-string
numsym   equ    stringlen/2

```

Пример 12. Переключение мерцание - яркость. Заполнить участок экрана надписями разного цвета. В цикле переключать значение бита мерцание - яркость.

```

;Определение
stdout   equ    1        ;Дескриптор стандартного вывода
;Основные фрагменты программы
;Инициализируем оба сегментных регистра DS и ES
...
;Очистим экран
        MOV AH,40h
        MOV BX,stdout
        MOV CX,homeLen

```

```

MOV DX,offset home
INT 21h
;Выведем строку line1 с позиции 12,10
MOV AH,13h          ;Функция вывода строки
MOV AL,0             ;Атрибут в BL
MOV BH,0             ;Страница 0
MOV BL,41h           ;Синий по красному
MOV CX,linelen       ;Длина строки
MOV DH,12            ;Координата Y
MOV DL,10            ;Координата X
MOV BP, offset line1 ;Адрес строки
INT 10h
;Выведем строку line2 с позиции 14,10
MOV AH,13h          ;Функция вывода строки
MOV AL,0             ;Атрибут в BL
MOV BH,0             ;Страница 0
MOV BL,0C1h         ;Синий по розовому
MOV CX,line          ;Длина строки
MOV DH,14            ;Координата Y
MOV DL,10            ;Координата X
MOV BP, offset line2 ;Адрес строки
INT 10h
;Выведем строку line3 с позиции 12,35
...
;Выведем строку line4 с позиции 14,35
...
;Включим мерцание
toggle: MOV AH,10h      ;Группа подфункций
        MOV AL,03h      ;Подфункция мерцание - яркость
        MOV BL,1        ;Мерцание
        INT 10h
;Введем задержку, заметно большую периода мерцания
        MOV CX,40
cxloop: MOV BX,0
delay:  INC BX
        JNE delay
        LOOP cxloop
;Включим яркость
        MOV AH,10h      ;Группа подфункций
        MOV AL,03h      ;Подфункция мерцание - яркость
        MOV BL,0        ;Яркость
        INT 10h
;Введем задержку, заметно большую периода мерцания
        MOV CX,40
cxloop1: MOV BX,0

```

```

delay1:    INC    BX
           JNE    delay1
           LOOP   cxloop1
;Анализ буфера клавиатуры функцией 06h прерывания DOS (INT 21h)
;и завершение программы по нажатию клавиши
...
           JZ     toggle           ;Z=0, символа нет
;Завершим программу
...
;Поля данных
home       db     27,'[2J' ;Очистка экрана
homelen    equ    $-home
line1      db     ' Синий по красному '
line2      db     ' Синий по розовому '
line3      db     ' Желтый по синему '
line4      db     ' Желтый по голубому '
linelen    equ    $-line4        ;Все строки одной длины

```

ОБОРУДОВАНИЕ

Объект исследования - программа на языке Ассемблера с использованием функций прерывания типа 21 операционной системы DOS. Средства исследования - программные модули Masm/Tasm, Link/Tlink, Debug/Turbo Debug, и IBM совмещенная ЭВМ.

ЗАДАНИЕ НА РАБОТУ

1. Изучить команды сравнения, перехода и прерывания.
2. Получить задание.
3. Написать программу ввода-вывода в соответствии с заданием.
4. Используя 21h прерывание ввести информацию в соответствии с заданием с клавиатуры, выполнить над ней необходимые преобразования и вывести ее на экран.

1	При вводе символа «1» на экран выводится символ «а»; при вводе «0» программа заканчивает свою работу; на остальные символы программа не реагирует
2	При вводе символа «2» на экран выводится символ «=»; при вводе символа «3» на экран выводится символ «?»; при вводе «0» программа заканчивает свою работу; на остальные символы программа не реагирует
3	Вывести на экран запрос и ввести строку – ответ на этот запрос
4	В зависимости от введенного символа («1» или «2») выдавать одно из двух сообщений
5	Нарисовать прямоугольник одинарной линией с сообщением внутри него
6	Нарисовать прямоугольник двойной линией с сообщением внутри него
7	Вывести сообщение в разные места экрана в зависимости от нажатой клавиши

8	Ввести два числа (от 0 до 9), сложить их, результат вывести на экран. ВНИМАНИЕ! Сумма должна быть меньше 10.
9	Ввести два числа (от 0 до 9). Разность вывести на экран. ВНИМАНИЕ! Первое число должно быть больше второго.
10	Ввести строку, обозначающую арифметический пример (например, 2+2=), проанализировать ее, выполнить введенную операцию, результат вывести на экран. ВНИМАНИЕ! Должны действовать ограничения вариантов 8 и 9.

5. Составить программу, позволяющую вывести средствами DOS на экран в текстовом режиме несколько строк текста с предварительной очисткой экрана.

	функ ция	вывод изображения	вывод следующей строки
1	40h	выделение яркостью	с позиции курсора
2	09h	выделение мерцанием	с позиции курсора
3	02h	инверсное изображение	с позиции курсора
4	06h	инверсное изображение	с позиции курсора
5	09h	выделение яркостью	после перемещения курсора на n строк вверх
6	40h	выделение мерцанием	после перемещения курсора на n строк вверх
7	02h	выделение мерцанием	после перемещения курсора на n строк вверх
8	06h	выделение яркостью	после перемещения курсора на n строк вверх
9	02h	выделение яркостью	после перемещения курсора на n строк вниз
10	09h	инверсное изображение	после перемещения курсора. на n строк вниз
11	06h	выделение мерцанием	после перемещения курсора на n строк вниз
12	40h	инверсное изображение	после перемещения курсора на n строк вниз

Дополнительно программа должна выполнять следующие задания:

	доп. требования	вид изображения
1	с установкой позиции курсора 5,7	выделение яркостью
2	с установкой позиции курсора 10,2	инверсное изображение
3	с установкой позиции курсора 11,6	синий фон ярко белые буквы
4	с установкой позиции курсора 2,10	белый фон темно синие буквы
5	с прокруткой вверх 3ст	инверсное изображение
6	с прокруткой вверх 1ст	выделение яркостью
7	с прокруткой вверх 2ст	инверсное изображение
8	с прокруткой вверх 4ст	выделение яркостью
9	с прокруткой вниз 3ст	синий фон ярко серые буквы
10	с прокруткой вниз 1ст	зеленый фон желтые буквы
11	с прокруткой вниз 2ст	выделение яркостью
12	с прокруткой вниз 4ст	инверсное изображение

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

5.1. Написать и отладить программы.

5.2. Оформить отчет.

6. ОФОРМЛЕНИЕ ОТЧЕТА

Отчет должен включать:

- 6.1. Вариант задания.
- 6.2. Текст программы.
- 6.3. Результаты работы программы.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

- 7.1. Что входит в функции контроллера клавиатуры?
- 7.2. Что такое "скен-код"?
- 7.3. Как работает программа обработки прерываний INT 09h?
- 7.4. Для чего служит кольцевой буфер ввода?
- 7.5. Что такое ввод с упреждением?
- 7.6. Что такое "расширенный код ASCII"?
- 7.7. Какие три уровня процедур ввода данных с клавиатуры предоставляет DOS?
- 7.8. Как осуществляется ввод с клавиатуры средствами файловой системы?
- 7.9. Перечислите группы прерываний системы BIOS.
- 7.10. Перечислите прерывания операционной системы DOS.
- 7.11. Какие функции для ввода информации прерывания типа 21 вы знаете?
- 7.12. Что означает функция A прерывания типа 21?
- 7.13. Какие регистры использует функция A?
- 7.14. Что означает функция 1 прерывания типа 21?
- 7.15. Как ввести строку символов с клавиатуры?
- 7.16. Что включает в себя видеосистема компьютера?
- 7.17. Сколько страниц хранится в памяти адаптера?
- 7.18. Какие функции DOS выводят текстовую информацию на экран?
- 7.19. Как перенаправить вывод информации?
- 7.20. Какие Esc-последовательности используются для управления экраном?
- 7.21. Какие функции драйвера BIOS (прерывание INT 10h) используются при работе в текстовом режиме?
- 7.22. Перечислите группы прерываний системы BIOS.
- 7.23. Перечислите прерывания операционной системы DOS.
- 7.24. Какие функции вывода информации на экран прерывания типа 21 вы знаете?
- 7.25. Как вывести строку символов на экран дисплея?