Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Московский государственный университет леса

А. В. Чернышов

УПРАВЛЕНИЕ РЕСУРСАМИ ЭВМ

Практикум
к выполнению лабораторных работ
Для студентов специальности 220100

Часть 2

Издательство Московского государственного университета леса
Москва — 2004

УДК 004.451.86(076.58):[004.33+004.35]
Ч-49

Ч-49 Чернышов А. В. Управление ресурсами ЭВМ: Практикум к выполнению лабораторных работ для студентов специальности 220100. Ч. 2. — М.: МГУЛ, 2004. — 32 с.

В практикуме даны необходимый минимум информации для выполнения лабораторных работ второго семестра по дисциплине «Операционные системы», а также индивидуальные задания на каждую лабораторную работу и требования к оформлению отчётов по ним.

Разработано в соответствии с Государственным образовательным стандартом ВПО 2000 г. для подготовки специалистов по направлению 654600 на основе примерной программы дисциплины «Операционные системы» для специальности 552800 (220100) «Информатика и вычислительная техника».

Одобрено и рекомендовано к изданию в качестве практикума редакционно-издательским советом университета

Рецензент — доцент В. Д. Кекин

Кафедра вычислительной техники

Автор — Александр Викторович Чернышов, доцент

ПРЕДИСЛОВИЕ

Настоящий практикум включает три лабораторные работы, которые должны быть выполнены студентами в течение второго семестра изучения дисциплины «Операционные системы», занимающей два учебных семестра.

Для выполнения работ студентам необходимо знание языков программирования Си и Ассемблер ПЭВМ IBM PC. Предполагается, что студенты изучают эти языки в рамках специальных курсов.

Лабораторные работы 1 и 3 построены таким образом, что студенты могут выполнять их в любой операционной системе. Рекомендуется выбрать одну из двух операционных систем (MS DOS или Linux), имеющихся в лабораторной аудитории.

Для выполнения лабораторной работы 2 необходим отдельный компьютер с подключённым матричным принтером и ОС MS DOS. Такой компьютер имеется в лабораторной аудитории.

К каждой лабораторной работе даётся необходимый минимум теоретического материала, позволяющий студентам уверенно выполнить работы 2 и 3 и получить общее представление о выполнении работы 1.

Достаточный для выполнения работы 1 материал студенты получают на лекционных занятиях курса «Операционные системы».

К каждой лабораторной работе для студентов одной группы подобраны по возможности индивидуальные задания. Однако все задания сформулированы не детально, что даёт возможность каждому студенту при написании конкретных программ проявить свою индивидуальность.

Лабораторная работа 1

МЕТОДЫ РАСПРЕДЕЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТИ

1. Общие сведения

Любая современная операционная система обеспечивает возможность распределения оперативной памяти между процессами с помощью двух системных вызовов: «выделить блок памяти» и «освободить блок памяти». При этом предполагается, что для выделения может быть запрошен блок памяти произвольного размера*, а освобождён только ранее выделенный блок памяти целиком.

1) Стратегии выделения памяти

Запрошенный блок памяти выделяется операционной системой из имеющейся в наличии непрерывной свободной области оперативной памяти. Как правило, при устоявшемся режиме работы системы свободная память представляет собой набор «дыр» некоторого размера среди блоков уже распределённой памяти. Таким образом, запрошенный блок должен по размеру быть не больше хотя бы одной «дыры».

Если в настоящий момент блок выделить невозможно, система может принять некоторые специальные меры: заблокировать запросивший память процесс до момента, пока станет доступной «дыра» подходящего размера; освободить несколько блоков оперативной памяти (с применением свопинга) для получения «дыры» подходящего размера; выполнить процедуру уплотнения памяти перемещением всех распределённых блоков в единую сплошную область со стремлением создать достаточно большую «дыру».

Рассмотрим только простейший случай, при котором запрошенный блок памяти выбирается из множества имеющихся «дыр». Все «дыры» обычно организованы в список, упорядоченный тем или иным способом. Если подходящая «дыра» не найдена, то процедура выделения памяти возвращает признак отказа.

---

* На практике запросы блоков памяти, превышающих физический размер оперативной памяти, бессмысленны.

Поиск подходящей «дыры» может быть реализован по одной из следующих стратегий: «первый подходящий», «следующий подходящий», «наилучший подходящий» и «наименее подходящий». Кроме того, при управлении оперативной памятью в системе может применяться так называемый «метод двойников», который будет рассмотрен отдельно.

При использовании стратегии «первый подходящий» ищут любую «дыру», такую, чтобы она была не меньше размеров запрошенного блока.

При стратегии «следующий подходящий» процесс поиска практически ничем не отличается от «первого подходящего» за тем исключением, что при выделении блока памяти запоминается место в списке «дыр», где был выделен блок, и в следующий раз поиск начинается не с начала списка, а с запомненного места (с обеспечением возможности после достижения конца списка просмотреть и его начало).

При стратегии «наилучший подходящий» выбирают ту «дыру», которая подходит по размерам наилучшим образом, то есть по размеру больше запрошенного блока на наименьшую величину из всех существующих «дыр».

При стратегии «наименее подходящий» в противоположность предыдущей выбирают «дыру» максимального размера.

Память может быть распределена как часть найденной «дыры» с уменьшением размеров самой «дыры» или как «дыра» целиком, если остающаяся нераспределённой часть «дыры» мала для того, чтобы иметь какую-либо ценность в будущем.

Как показывает практика и специально проведённое сравнительное моделирование, в общем случае ни одна из перечисленных стратегий не имеет решающего преимущества над другой в смысле обеспечения возможности выделения запрашиваемых блоков оперативной памяти в устоявшемся режиме работы системы. Но стратегия «первый подходящий» работает в общем значительно быстрее всех остальных. Поэтому на практике, если нет особых предпосылок к бóльшей эффективности другой стратегии, чаще всего применяют её.

Может показаться, что достаточно упорядочить список «дыр» по возрастанию (убыванию) их размеров, и стратегии «наилучший подходящий» или «наименее подходящий» практически сравняются по скорости поиска нужной «дыры» со стратегией «первый подходящий». Однако при этом возникнет другая проблема, связанная с поиском смежных «дыр» при их объединении.

2) Освобождение памяти

Когда блок памяти становится не нужен, он возвращается операционной системе, и в памяти появляется новая «дыра». При этом, если эта «дыра» оказывается смежной с уже существующей «дырой», то необходимо объединить две «дыры» в одну «дыру» бóльшего размера. Эта задача может оказаться достаточно сложной на практике.

Предположим, что список «дыр» хранится в виде простого односвязного списка (рис. 1.1,*а*). При освобождении памяти наиболее простым является добавление информации о новых «дырах» к одному из концов списка (обычно к началу). Поэтому, если в начальный момент времени список был упорядочен по возрастанию адресов, то с течением времени упорядоченность будет естественным образом нарушена. Можно, конечно, принять специальные меры по поддержанию упорядоченности списка (рис. 1.1,*б*), но это потребует проведения поиска в списке места, куда должен быть помещён освобождённый блок, то есть дополнительных накладных расходов.

Для эффективного решения задачи объединения «дыр» должны использоваться соответствующие структуры данных.

Обычно невозможно по отдельности решать задачи поиска подходящей «дыры» при выделении блока памяти и поиска смежных «дыр» для их слияния при освобождении блока памяти. Поэтому и структуры данных, используемые для управления оперативной памятью, должны учитывать решение обеих задач сразу.

3) Структуры данных для управления распределением памяти

В простейшем случае память может быть поделена на блоки фиксированного (небольшого) размера. В этом случае в ответ на запрос выделяется непрерывная последовательность блоков, достаточная по суммарному объёму. Для хранения информации о занятых и свободных блоках памяти используется так называемая битовая карта памяти, в которой каждому биту соответствует свой блок памяти. Если блок занят, бит равен единице, иначе — нулю.

В более общем случае используется выделение блоков памяти произвольного размера, соответствующих запросу. При этом информацию о свободных блоках памяти хранят в самих блоках, размещая её как запись в начале каждого свободного блока («дыры») и организуя эти записи в виде списка (см. рис. 1.1). Каждая запись содержит информацию о размере блока и указатель на следующий блок.

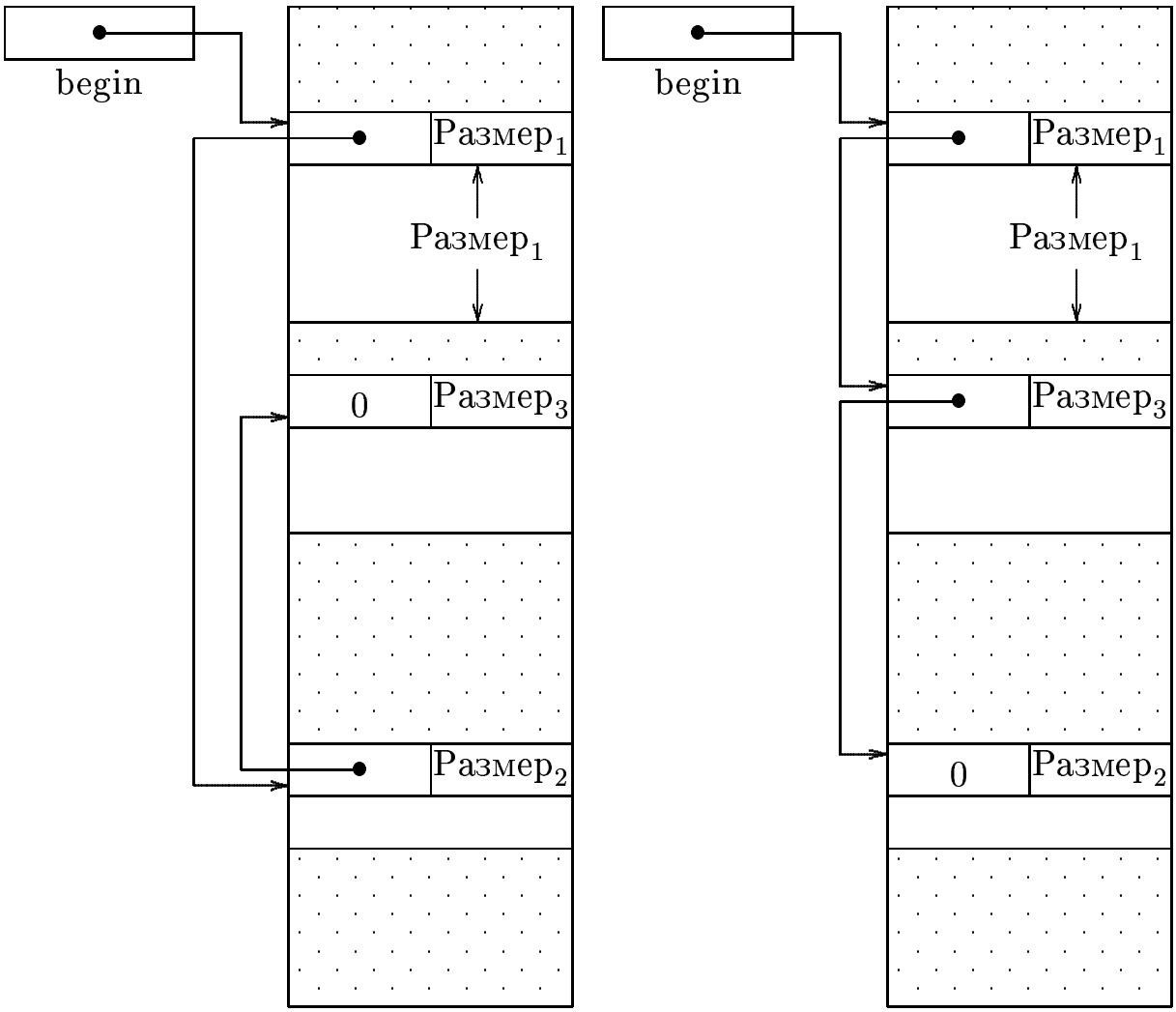

Рис. 1.1. Хранение информации о «дырах» в виде односвязного списка: *а* — неупорядоченного; *б* — упорядоченного по возрастанию адресов памяти; заштрихованы уже распределённые блоки памяти

Как уже было показано, при такой организации структур данных для определения сопряжённых «дыр» удобнее всего поддерживать список блоков упорядоченным по адресам памяти, то есть добавлять освобождаемые блоки не в какой-либо конец списка, а в середину, в соответствии с их адресами.

Усложнив структуры данных, можно избавиться от упорядочивания по адресам. При этом создаётся двусвязный список свободных блоков, а каждый блок в начале и в конце снабжается специальными «этикетками», которые содержат информацию о том занят блок или свободен (рис. 1.2). При освобождении блок может быть записан в любое место списка. Проверка на сопряжённые «дыры» выполняется по расположенным в соседних ячейках памяти «этикеткам».

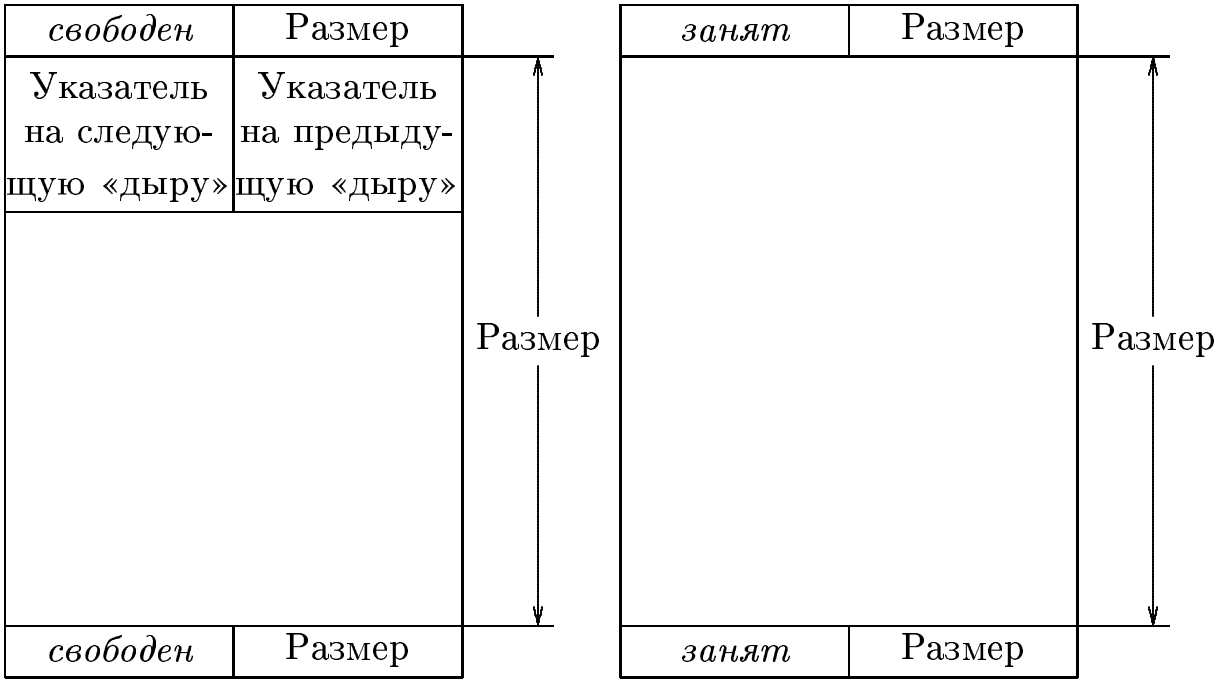

Рис. 1.2. Блок данных — элемент двусвязного списка с «этикетками»: *а* — свободный («дыра»); *б* — распределённый (занятый)

4) Метод двойников

При этом методе рассматривается оперативная память объёмом 2^N и минимально допустимый для выделения блок этой памяти объёмом 2^M, $(M < N)$. Например, при $N = 10$ и $M = 4$ имеем максимальный объём оперативной памяти $2^{10} = 1024$ байта и минимальный объём блока $2^4 = 16$ байт.

При запросе блока памяти размером s такого, что $2^{i-1} < s \leqslant 2^i$, где $M < i \leqslant N$ реально будет выделен блок размером 2^i байт. При этом идея процесса поиска блока состоит в следующем:

а) первоначально вся память размером 2^N свободна;

б) при запросе блока памяти размером s память делится пополам с образованием блоков-двойников равного размера (2^i, где для первой итерации $i = N - 1$);

в) если $s > 2^i$, то образовавшиеся двойники сливаются обратно и получившийся блок памяти выделяется по запросу. Если $s = 2^i$, то по запросу выделяется один из образовавшихся двойников. Второй при этом остаётся свободным и может быть использован для выделения по другому запросу. Иначе ($s < 2^i$ и $i > M$) выбирается один из блоков-двойников и с ним повторяются операции (б) и (в);

г) если $i = M$, то по запросу выделяется блок размером 2^M. Дальнейшего деления не происходит.

Образовавшиеся в процессе выделения блока памяти двойники сохраняют свою индивидуальность вплоть до момента освобождения обоих двойников. Лишь после этого они снова сливаются, образуя единый блок (который в свою очередь является двойником другого блока такого же размера).

Система должна вести учёт свободных и занятых двойников. В устоявшемся режиме работы поиск блока подходящего размера нужно начинать с наименьшего свободного блока-двойника, бóльшего запрошенного размера.

Подробнее о структурах данных и стратегиях распределения памяти см. [1], с. 488–507.

2. Задание на лабораторную работу

Написать на языке Си программу, демонстрирующую один из способов распределения памяти, выбираемый студентом из табл. 1.1 в соответствии со своим номером в журнале. Программа должна в начале работы запросить большой блок памяти (пул) у операционной системы, а затем применять собственные (разработанные студентом) процедуры распределения памяти для выделения и освобождения блоков памяти внутри запрошенного пула.

Процедурами распределения памяти считаются процедуры выделения блока памяти заданного размера и освобождения ранее выделенного блока памяти. Процедура выделения блока памяти должна возвращать спецификатор начала выделенного блока. Процедура освобождения блока памяти должна освобождать блок, заданный спецификатором. Результаты работы программы должны быть представлены в печатной форме и внесены в отчёт.

3. Содержание отчёта

Отчёт должен содержать:

– титульный лист;

– задание на работу;

– краткое описание используемых алгоритмов распределения памяти;

– исходный текст процедур управления памятью и демонстрационной программы;

– результаты работы демонстрационной программы.

Т а б л и ц а 1.1

Номер по журналу	Стратегия размещения	Структура данных	Избегать малых «дыр»
1	$\rightarrow 1$	#	Нет
2	$\mapsto$	$\hookrightarrow$	Нет
3	$\overset{\cdot\cdot}{\smile}$	$\Leftrightarrow$	Нет
4	$\overset{\cdot\cdot}{\frown}$	#	Нет
5	$\rightarrow 1$	$\hookrightarrow$	Нет
6	$\mapsto$	$\Leftrightarrow$	Нет
7	$\overset{\cdot\cdot}{\smile}$	#	Нет
8	$\overset{\cdot\cdot}{\frown}$	$\hookrightarrow$	Нет
9	$\rightarrow 1$	$\Leftrightarrow$	Нет
10	$\mapsto$	#	Нет
11	$\overset{\cdot\cdot}{\smile}$	$\hookrightarrow$	Нет
12	$\overset{\cdot\cdot}{\frown}$	$\Leftrightarrow$	Нет
13	$\rightarrow 1$	#	Да
14	$\mapsto$	$\hookrightarrow$	Да
15	$\overset{\cdot\cdot}{\smile}$	$\Leftrightarrow$	Да
16	$\overset{\cdot\cdot}{\frown}$	#	Да
17	$\rightarrow 1$	$\hookrightarrow$	Да
18	$\mapsto$	$\Leftrightarrow$	Да
19	$\overset{\cdot\cdot}{\smile}$	#	Да
20	$\overset{\cdot\cdot}{\frown}$	$\hookrightarrow$	Да
21	$\rightarrow 1$	$\Leftrightarrow$	Да
22	$\mapsto$	#	Да
23	$\overset{\cdot\cdot}{\smile}$	$\hookrightarrow$	Да
24	$\overset{\cdot\cdot}{\frown}$	$\Leftrightarrow$	Да
25	$\bowtie$	–	–

П р и м е ч а н и е. Условные обозначения: $\rightarrow 1$ — «Первый подходящий»; $\mapsto$ — «Следующий подходящий»; $\overset{\cdot\cdot}{\smile}$ — «Наилучший подходящий»; $\overset{\cdot\cdot}{\frown}$ — «Наименее подходящий»; $\bowtie$ — «Метод двойников»; # — Блоки фиксированного размера; $\hookrightarrow$ — Односвязный упорядоченный список; $\Leftrightarrow$ — Двусвязный список с этикетками.

4. Навыки, полученные студентом

После выполнения лабораторной работы студент должен:

– иметь представление о способах распределения памяти, применяемых в вычислительных системах;

– знать основные структуры данных для управления памятью и

стратегии распределения памяти;

– владеть методами выбора необходимых структур и стратегий;

– уметь практически реализовать процедуры управления памятью.

Лабораторная работа 2

УПРАВЛЕНИЕ ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

1. Общие сведения

Одной из важных задач операционной системы является обеспечение интерфейса между программами пользователей и различными периферийными устройствами, с которыми эти программы должны взаимодействовать в процессе выполнения.

Каждое периферийное устройство подключено к ЭВМ с помощью какого-либо аппаратного контроллера или схемы сопряжения. Такой контроллер позволяет центральному процессору или каналу ввода-вывода осуществлять обмен данными с устройством через выделенные порты ввода-вывода. При этом часть передаваемых через порты данных может управлять режимами самого контроллера или обеспечивать протокол обмена контроллера и периферийного устройства. И только оставшаяся часть данных будет представлять собственно данные, предназначенные для периферийного устройства.

Однако и в массиве данных, «дошедших» до устройства, обычно лишь часть представляет собой собственно данные, а остальное — команды управления режимами работы данного периферийного устройства.

Естественно, что операционная система должна, по-возможности, избавлять прикладные программы (и в конечном счёте пользователей и программистов) от необходимости учитывать все особенности и самих периферийных устройств, и контроллеров, посредством которых они подключены к ЭВМ. Прикладные программы должны иметь возможность передавать данные на периферийные устройства посредством простых запросов. Если же возникает необходимость в переключении режима работы периферийного устройства, то прикладная программа должна иметь возможность сделать это путём стандартного запроса к операционной системе, либо передачей на устройство простой командной последовательности, не зависящей от аппаратных особенностей конкретной модели устройства.

Типичным примером периферийного устройства, имеющего описанные особенности, является принтер. В настоящей работе будет рассмотрено управление матричным 9-игольчатым принтером. Обычно такой принтер подключён к компьютеру через параллельный контроллер ввода-вывода, имеющий обобщённое название «порт Centronics».

Для передачи на принтер, подключённый к компьютеру, одного байта процессор должен выполнить ряд манипуляций с портами ввода-вывода контроллера.

Сам принтер не только печатает полученные от компьютера байты. В потоке байтов он распознаёт управляющие последовательности, которые предназначены для управления его работой. Посредством управляющих последовательностей принтеру могут быть переданы команды на смену шрифта, формата бумаги, межстрочного интервала, режима печати, и др.

Строго говоря, на сегодняшний день существует два типа принтеров, отличающихся способами управления. К первому относятся устройства, которые работают по только что описанному принципу, то есть в основном просто печатают полученные байты и лишь некоторые их последовательности воспринимают как управляющие команды.

Ко второму типу устройств относятся так называемые PostScript-принтеры. Эти устройства весь переданный для них поток данных воспринимают как программу печати, в процессе выполнения которой на бумаге формируется необходимое изображение (в том числе и обычный текст). Этот тип принтеров в настоящей работе не рассматривается в силу сложности управляющего языка.

1) Управление контроллером принтера

а) Описание портов ввода-вывода

Для управления контроллером сопряжения с принтером в ПЭВМ IBM PC выделены три порта ввода-вывода: 378h, 379h и 37Ah*.

Порт 378h доступен на запись и чтение и предназначен для записи очередного байта, передаваемого на принтер. При чтении порта можно прочитать только что записанный байт.

Порт 379h доступен только для чтения и позволяет получить текущее состояние принтера. Значения битов порта приведены в табл. 2.1.

---

* Речь идёт о принтере, подключённом к первому контроллеру. В общем случае возможно подключение до трёх печатающих устройств (не считая подключённых по последовательным линиям). Но типичный компьютер IBM PC обычно имеет только один контроллер принтера.

Т а б л и ц а 2.1

Номер бита	Назначение бита
0	Не используется (равно 0)
1	Не используется (равно 0)
2	Не используется (равно 0)
3	Сигнал ошибки, активный уровень 0
4	Принтер выбран
5	Конец бумаги
6	0 — Готовность принтера
7	0 — Принтер занят, находится в OffLine или произошла ошибка

Т а б л и ц а 2.2

Номер бита	Назначение бита
0	Строб данных, принимает значение 1 при выводе байта
1	Автоматический перевод строки после символа «возврат каретки»
2	Сброс принтера (активный уровень 0)
3	Выбор принтера для работы
4	Разрешение прерывания от принтера (прерывание IRQ7, INT 0Fh)
5	Не используется (равно 0)
6	Не используется (равно 0)
7	Не используется (равно 0)

Порт 37Ah доступен для чтения и записи и позволяет управлять процессом передачи байтов на принтер и частично самим принтером. Значения битов порта приведены в табл. 2.2.

Если разрешено прерывание от принтера, то оно происходит всякий раз, когда принтер готов принять очередной байт данных.

б) Описание прерываний BIOS

Для работы с принтером в BIOS предусмотрены функции 0, 1, 2 прерывания 17h.

Функция 0 — выводит на принтер один символ, заданный в регистре AL.

Вход:
AH=00h
AL=код символа для печати
DX=номер принтера (обычно 0)
Выход:
AH=слово состояния принтера (табл. 2.3).

Т а б л и ц а 2.3

Номер бита	Назначение бита
0	Тайм-аут при выводе байта (байт не выведен)
1	Не используется (равно 0)
2	Не используется (равно 0)
3	Ошибка ввода-вывода
4	1 — Принтер выбран; 0 — Принтер в состоянии OffLine
5	Конец бумаги
6	Подтверждение
7	0 — Принтер занят; 1 — Принтер готов

Функция 1 — инициализация принтера (аппаратный сброс в режим по умолчанию).
Вход:
AH=01h
DX=номер принтера (обычно 0)
Выход:
AH=слово состояния принтера (см. табл. 2.3).
Функция 2 — получить текущее состояние принтера.
Вход:
AH=02h
DX=номер принтера (обычно 0)
Выход:
AH=слово состояния принтера (см. табл. 2.3).

в) Описание прерываний DOS

Операционная система MS DOS предоставляет специфический сервис обслуживания процесса печати, фактически представляющий собой передачу байтов в стандартный файл печати LPT1 (PRN). Сервис вызывается как функция 5 прерывания 21h.

Вход:

AH=05h

DL=код символа для печати

Возврат: нет. Вместо этого при ошибке вывода вызывается стандартный обработчик ошибок, на экран выводится сообщение об ошибке записи в порт принтера и пользователю предлагается принять решение, пытаться ли дальше передавать данные или прекратить работу программы.

2) Управляющие последовательности принтера

Управление работой матричных принтеров выполняется с помощью так называемых ESC-последовательностей, получивших своё название за то, что большинство из них начинаются с символа ESC (033_8, 27_{10}, $1Bh_{16}$). Разные модели матричных принтеров поддерживают немного различающиеся последовательности. Часть этих последовательностей выполняет специфические функции, редко применяемые на практике. Поэтому здесь приведём лишь наиболее известные и полезные на практике последовательности.

В настоящем описании приняты следующие соглашения:

ESC — символ ESC (байт с кодом 033_8, 27_{10}, $1Bh_{16}$);

n — некоторый байт с кодом n;

(13) — байт с десятичным кодом 13_{10} (в данном случае код возврата каретки);

'n' — символ 'n'.

`ESC '@'` — инициализация принтера.

Выполняется сброс в состояние по конфигурации.

`(13)` — Возврат каретки.

Команда вызывает распечатку содержимого буфера данных принтера, после чего головка переводится в начало текущей строки.

`(10)` — Перевод строки.

Команда вызывает распечатку содержимого буфера данных принтера, после чего бумага подаётся вперёд на текущую величину межстрочного интервала. Печатающая головка остаётся в текущей позиции, если только на принтере не включён режим автоматического возврата каретки в начало строки.

(12) — Подача бумаги на один лист.

Команда вызывает распечатку содержимого буфера принтера, после чего происходит подача бумаги вперед на один лист (в соответствии с установленной длиной листа).

ESC 'C' n — Установить длину листа в строках.

Устанавливается длина листа n строк. При этом используется ранее установленное расстояние между строками. $1 \leqslant n \leqslant 127$.

ESC 'C' 0 n — Установить длину листа бумаги в дюймах.

Длина листа бумаги устанавливается в n дюймов. $1 \leqslant n \leqslant 22$.

ESC 'C' '0' n — Установить длину листа бумаги в дюймах.

Длина листа бумаги устанавливается в n дюймов. $1 \leqslant n \leqslant 22$.

ESC 'N' n — Установить дополнительный пропуск между страницами в n строка. $1 \leqslant n \leqslant 127$.

ESC 'O' — Отменить дополнительный пропуск между страницами, установленный командой ESC 'N'.

ESC '0' — Выбор межстрочного интервала 1/8".

ESC '1' — Выбор межстрочного интервала 7/72".

ESC '2' — Выбор межстрочного интервала 1/6".

ESC '3' n — Выбор межстрочного интервала, равного n/216". $0 \leqslant n \leqslant 255$.

ESC 'A' n — Выбор межстрочного интервала, равного n/72". $0 \leqslant n \leqslant 85$.

ESC 'J' n — Проброс бумаги вперёд на расстояние n/216".

Команда выполняется немедленно и не вызывает перемещения печатающей головки.

ESC 'j' n — Проброс бумаги назад на расстояние n/216".

Команда выполняется немедленно и не вызывает перемещения печатающей головки. К сожалению, команда не выполняется на многих старых моделях принтеров.

ESC 'I' n — Установка левой границы.

Левая граница листа устанавливается в n символов текущей ширины.

(8) — Возврат на одну позицию.

(9) — Горизонтальная табуляция.

Вообще говоря, существует специальная команда задания позиций горизонтальной табуляции. Но по умолчанию можно считать, что эти границы всегда соответствуют номерам позиций, кратным 8.

ESC 'x' n — выбор черновой или качественной печати.

Параметр n определяет режим печати:

0 или '0' — черновой режим;

1 или '1' — качественный режим.

ESC 'P' — Выбор плотности печати в 10 символов на дюйм.

ESC 'M' — Выбор плотности печати в 12 символов на дюйм.

(15) — Выбор режима плотной печати.

ESC (15) — Выбор режима плотной печати.

(18) — Отмена режима плотной печати.

(14) — Включить печать с двойной шириной.

Этот режим отменяется по возврату каретки или по (20).

ESC (14) — Включить печать с двойной шириной.

Команда аналогична команде (14).

(20) — Отмена режима печати с двойной шириной.

Команда не действует на режимы, включенные по ESC 'w' или ESC '!'.

ESC 'W' n — Включение/выключение режима печати с двойной высотой.

Значения n:

0 или '0' — выключение режима;

1 или '1' — включение режима.

ESC 'w' n — Включение/выключение режима печати с двойной шириной.

Значения n:

0 или '0' — выключение режима;

1 или '1' — включение режима.

ESC 'E' — установка режима печати с выделением.

ESC 'F' — выключение режима печати с выделением.

ESC 'G' — Установка режима двойной пропечатки.

`ESC 'H'` — Выключение двойной пропечатки.

`ESC 'S' n` — Печать индексами.
Значения n:
0 или '0' — верхний индекс;
1 или '1' — нижний индекс.

`ESC 'T'` — Выключение печати индексами.

`ESC '-' n` — Включение/выключение режима подчеркивания.
Значения n:
0 или '0' — выключение режима;
1 или '1' — включение режима.

`ESC '_' n` — Включение/выключение режима перечеркивания.
В зависимости от значения параметра n все символы печатаются перечеркнутыми или нет.
Значения n:
0 или '0' — выключение режима;
1 или '1' — включение режима.

`ESC 'пробел' n` — Установка дополнительного расстояния между символами.
Параметр n задает количество точек, добавляемых справа к каждому символу.

`ESC '4'` — Включить печать курсивом.

`ESC '5'` — Выключить печать курсивом.

`ESC '/'` n_1 n_2 — Печать символов с кодами, меньшими (32).
Разрешается печать следующих за командой $(n_2 \times 256) + n_1$ символов, имеющих коды меньше (32).

`ESC '^'` — Разрешается печать одного следующего за командой символа с кодом меньше (32).

`ESC '!' n` — Выбор режима работы принтера.
Отдельные биты байта n задают режимы работы принтера, описанные в табл. 2.4.

`ESC '~' n` — Включение/выключение инверсного режима печати.
Значения n:
0 или '0' — выключение режима;
1 или '1' — включение режима.

Таблица 2.4

Номер бита	Назначение бита
0	Плотность печати, символов на дюйм: 0 — 10; 1 — 12
1	Пропорциональный шрифт
2	Плотный шрифт
3	Выделенный шрифт
4	Двухпроходная печать
5	Двойная высота
6	Курсив
7	Подчёркивание

`ESC 'K'` n_1 n_2 — Включение графического режима с одинарной плотностью (60 точек на дюйм).

Печатается $(n_2 \times 256) + n_1$ графических столбцов по 8 бит, следующих за командой, в режиме одинарной плотности.

`ESC 'L'` n_1 n_2 — Включение графического режима с двойной плотностью (120 точек на дюйм).

Печатается $(n_2 \times 256) + n_1$ графических столбцов по 8 бит, следующих за командой, в режиме двойной плотности.

`ESC 'Y'` n_1 n_2 — Включение графического режима с двойной плотностью (120 точек на дюйм) и высокой скоростью печати.

Печатается $(n_2 \times 256) + n_1$ графических столбцов по 8 бит, следующих за командой, в режиме двойной плотности. При этом соседние по горизонтали точки не пропечатываются.

`ESC 'Z'` n_1 n_2 — Включение графического режима с учетверённой плотностью (240 точек на дюйм).

Печатается $(n_2 \times 256) + n_1$ графических столбцов по 8 бит, следующих за командой, в режиме учетверённой плотности.

2. Задание на лабораторную работу

Написать на языке Си программу, обеспечивающую печать на матричном 9-игольчатом принтере, подключённом к ПЭВМ, текстового файла, содержащего печатный текст и простые команды управления режимами печати этого текста. Предусмотреть следующие группы команд управления:

– переключение шрифтов;

– печать специальных символов;

– печать верхних и нижних индексов.

В качестве команд переключения шрифтов распознавать последовательности символов: #r — нормальный шрифт, #b — полужирный шрифт, #i — курсив. При этом реализовать их по одному из вариантов:

а) с помощью ESC-последовательностей принтера;

б) имитацией, когда полужирный шрифт реализуется как тройная пропечатка каждой буквы, а вместо курсива используется имитация подчёркивания (печать буквы и на том же месте символа подчёркивания).

В качестве команд печати специальных символов распознавать последовательности символов: #a — угол (∠), #o — жирная точка (•), #q — чёрный квадрат (▪). При этом реализовать их по одному из вариантов:

в) имитацией, накладывая друг на друга два-три подходящих стандартных символа (например, угол — подчёркивание и слеш, жирная точка — буква «о» и буква «х» и т. п.);

г) печатью графических вставок, воспроизводящих необходимые символы.

В качестве команд печати верхних и нижних индексов распознавать последовательности символов: #u — верхний индекс, #l — нижний индекс, #n — нормальный текст (например, последовательность x_i^2 может быть задана как «x#li#u2#n». При этом реализовать их по одному из вариантов:

д) ESC-последовательностями включения/выключения печати индексов;

е) ESC-последовательностями проброса бумаги.

Кроме того, для посылки данных на принтер использовать один из вариантов:

ж) прямое обращение к портам ввода-вывода;

з) использование сервиса BIOS;

и) использование сервиса DOS.

Набор конкретных вариантов для выполнения лабораторной работы представлен в табл. 2.5.

Т а б л и ц а 2.5

Номер по журналу	Управление шрифтами	Печать символов	Печать индексов	Посылка данных
1	а	в	д	ж
2	б	г	д	з
3	а	г	д	и
4	б	в	д	ж
5	а	в	д	з
6	б	г	д	и
7	а	г	д	ж
8	б	в	д	з
9	а	в	д	и
10	б	г	д	ж
11	а	г	д	з
12	б	в	д	и
13	а	в	е	ж
14	б	г	е	з
15	а	г	е	и
16	б	в	е	ж
17	а	в	е	з
18	б	г	е	и
19	а	г	е	ж
20	б	в	е	з
21	а	в	е	и
22	б	г	е	ж
23	а	г	е	з
24	б	в	е	и
25	а	в	е	ж

3. Содержание отчёта

Отчёт должен содержать:

– титульный лист;

– задание на работу;

– краткое описание используемых аппаратных ресурсов ПЭВМ и принтера, идей и алгоритмов, использованных при решении поставленной задачи;

– исходный текст программы печати;

– текст тестового файла для распечатки (файл должен содержать не менее 15 строк текста и команды, демонстрирующие печать во всех заданных режимах);

– результаты работы демонстрационной программы (результат респечатки).

4. Навыки, полученные студентом

После выполнения лабораторной работы студент должен:

– иметь представление о способах управления периферийными устройствами ПЭВМ;

– знать наиболее часто встречающиеся ESC-последовательности матричных принтеров;

– владеть методами выбора необходимых способов печати;

– уметь практически реализовать программу управления принтером.

Лабораторная работа 3

МОДЕЛИРОВАНИЕ РАБОТЫ ДИСКОВОЙ ПОДСИСТЕМЫ

1. Общие сведения

Операции ввода-вывода данных являются обязательной частью любой прикладной программы. А так как эти операции выполняются значительно медленнее, чем операции чтения и записи данных процессором в оперативной памяти, то влияние их на общую производительность вычислительной системы очень велико.

Для однозадачной операционной системы (ОС) с большой длительностью операций ввода-вывода приходится обычно мириться, как с неизбежным злом. Существующие методы кеширования записи на диск позволяют решить проблему увеличения скорости работы программы лишь частично, поскольку при объёме передаваемых данных, сравнимым с размерами кеша или превосходящим его, процесс фактически сводится к выполнению операций ввода-вывода непосредственно на устройство.

Для многозадачных же ОС разработано несколько методов, позволяющих повысить производительность вычислительной установки в целом. Необходимо однако понимать, что речь идёт прежде всего именно о производительности в целом, в отношении суммарного объёма вычислительной работы, выполняемой в единицу времени. Это суммарное повышение производительности возможно потому, что пока один процесс ожидает окончания своей операции ввода-вывода, другой процесс может продолжать выполнение на процессоре.

Наибольший интерес представляет изучение процесса ввода-вывода на дисковые устройства (гибкие и жёсткие магнитные диски). В силу технических особенностей своей конструкции эти устройства при использовании в многозадачных системах могут сильно влиять на суммарную производительность вычислительной системы. Дело в том, что время выполнения одного запроса t можно представить следующим выражением:

$$t = t_p + t_o + t_{rw},$$

где t_p — время поиска (подвода головки к нужному цилиндру);
t_o — время ожидания, пока нужный сектор диска не окажется под головкой;
t_{rw} — время собственно чтения или записи сектора.

В общем объёме времени t величина t_{rw} мизерна и практически не может быть изменена программными методами. Величина t_o никогда не превосходит по времени периода вращения диска и, хотя и может быть незначительно оптимизирована, выигрыш от такой оптимизации бывает ощутимым только в сильно загруженных вычислительных системах.

Напротив, величина t_p обычно на порядок или даже два может превосходить и t_o, и t_{rw}. И именно её уменьшение представляет наибольший интерес.

Оптимизация t_p возможна лишь в том случае, если имеется несколько процессов, генерирующих запросы к дисковому устройству (причём к разным его частям) в случайные моменты времени. То есть в некоторые моменты времени может поступать сразу несколько запросов, а в другие моменты времени запросов нет вовсе. Ещё раз подчеркнём, что оптимизируется производительность всей системы, а не скорость выполнения каждого запроса. Если в общем (в среднем) время выполнения запросов уменьшается (и увеличивается количество обслуженных запросов в единицу времени), то для отдельных запросов время обслуживания может значительно вырасти.

Обслуживание поступающих запросов на ввод-вывод может быть реализовано в ОС по одной из стратегий, описанных далее.

1) Стратегия FIFO

Эта стратегия реализует принцип «первый пришёл — первым обслужен».

В этой стратегии запросы обслуживаются в порядке поступления. Фактически это есть стратегия без оптимизации. Её сильной стороной является «справедливость» процедуры обслуживания.

Такая стратегия может быть эффективна в ОС с малой нагрузкой на дисковую подсистему, так как исключает необходимость дополнительного планирования.

При средней и большой нагрузке такая стратегия неэффективна. Её характеристики будут использоваться в настоящей лабораторной работе как точка отсчёта для сравнения с другими стратегиями оптимизации.

2) Стратегия SSTF

Принцип этой стратегии «с наименьшим временем поиска — первым».

Основная идея стратегии заключается в том, что среди всех поступивших к данному моменту запросов к диску выбирается на обслуживание тот запрос, чей цилиндр находится ближе всего к текущему положению головки.

После обслуживания запроса снова просматриваются все поступившие запросы (включая новые) и снова выбирается «ближайший». И этот процесс циклически повторяется.

Стратегия имеет тенденцию к преимущественному обслуживанию запросов, группирующихся на рядом расположенных дорожках, и может приводить к бесконечному откладыванию «далёких» запросов.

3) Стратегия SCAN

В этой стратегии головка совершает сканирующие движения над поверхностью диска в одну и в другую сторону, обслуживая по пути все встретившиеся запросы. Смена направления происходит только тогда, когда в текущем направлении движения больше нет запросов для обслуживания.

Стратегия имеет тенденцию к лучшему обслуживанию запросов, относящихся к средним цилиндрам.

4) Стратегия C-SCAN

В этой стратегии головка совершает циклические сканирующие движения, а именно: движется в одном направлении (от края к центру), обслуживая все запросы по пути, а дойдя до самого внутреннего запроса, скачком возвращается к самому внешнему запросу и повторяет цикл сканирования.

5) Стратегия FSCAN

Это так называемое «шаговое сканирование». Головка движется также как и в стратегии SCAN, но обслуживаются только те запросы, которые поступили до начала движения головки в данном направлении. Все вновь поступившие запросы переупорядочиваются для обслуживания во время обратного движения головки.

Стратегия преодолевает проблему «бесконечного откладывания», возможную в стратегии SCAN, если в системе поступает большое количество запросов к какому-либо одному цилиндру.

6) Стратегия N-Step SCAN

Это так называемое «N-шаговое сканирование». Головка движется также как и в стратегии SCAN, но обслуживаются только те запросы, которые поступили до начала движения головки в данном направлении и не превысили заданное количество N. Все остальные запросы, включая вновь поступившие, количество которых на превышает величины N, переупорядочиваются для обслуживания во время обратного движения головки. И так далее.

Стратегия преодолевает проблему «бесконечного откладывания», возможную в стратегии SCAN, если в системе поступает большое количество запросов к какому-либо одному цилиндру.

7) Схема Эшенбаха

Стратегия аналогична C-SCAN с тем лишь отличием, что учитывается расположение запросов на каждом цилиндре. При движении в каждом направлении выполняется обслуживание только таких запросов, которые находятся на неперекрывающихся секторах каждого цилиндра, то есть могут быть выполнены за один оборот диска. Остальные запросы откладываются до следующего прохода.

2. Задание на лабораторную работу

Написать на языке Си программу, выполняющую моделирование работы дисковой подсистемы. (Реально обращения к диску выполняться не должны!) В качестве исходных данных использовать данные из табл. 3.1, соответствующие своему номеру в журнале, а также следующие:

– геометрия диска: 500 дорожек (цилиндров), 4 поверхности/головки, 16 секторов на поверхности;

– время перехода головки с дорожки на соседнюю дорожку: 0,5 мс;

– скорость вращения диска: 10 000 об/мин;

– время чтения одного сектора равно времени прохождения сектора под головкой (время ожидания сектора вычислять, исходя из номера запрошенного сектора, числа пересекаемых головкой дорожек и скорости вращения диска);

– время записи одного сектора складывается из времени прохождения сектора под головкой для собственно записи и повторного прохождения для верификации;

– моменты времени появления запросов к диску определять с использованием генератора случайных чисел (каждый вызов генератора даёт интервал времени от текущего запроса до следующего). При этом максимальное числовое значение, которое может выдать генератор должно соответствовать интервалу времени $t_{\max}$ с (см. табл. 3.1);

– каждый запрос к диску должен описываться следующими параметрами: номер цилиндра (дорожки), номер головки (поверхности), номер сектора, тип операции (чтение или запись), задаваемыми случайным образом;

– количество последовательных секторов, обрабатываемых в каждом запросе, должно задаваться случайным образом в пределах от 1 до n (см. табл. 3.1).

Программа должна выполнить моделирование для стратегии FIFO и стратегии, указанной в табл. 3.1. При этом моделирование должно быть выполнено для идентичных потоков запросов на интервале модельного времени 5 мин.

В качестве результатов моделирования должны быть получены величины: минимальное, максимальное и среднее время обслуживания запроса, среднеквадратическое отклонение от среднего, максимальная длина очереди запросов, суммарное время простоя дисковой подсистемы (отсутствие запросов). Построить гистограмму распределения времени обслуживания запросов.

С помощью разработанной моделирующей программы оценить возможности моделируемых стратегий для следующих значений $t_{\max}$: заданного; $t_{\max}/10$; $t_{\max}/100$.

3. Содержание отчёта

Отчёт должен содержать:

– титульный лист;

– задание на работу;

– краткое описание моделируемых стратегий обслуживания запросов и их реализации в программе;

– исходный текст программы;

– результаты моделирования;

– краткие выводы по результатам моделирования.

Т а б л и ц а 3.1

Номер по журналу	Стратегия обслуживания	Параметр $t_{\max}$, с	Параметр n
1	SSTF	2	1
2	SCAN	2	1
3	C-SCAN	2	1
4	FSCAN	2	1
5	N-Step	2	1
6	Эшенбах	2	1
7	SSTF	5	1
8	SCAN	5	1
9	C-SCAN	5	1
10	FSCAN	5	1
11	N-Step	5	1
12	Эшенбах	5	1
13	SSTF	2	16
14	SCAN	2	16
15	C-SCAN	2	16
16	FSCAN	2	16
17	N-Step	2	16
18	Эшенбах	2	16
19	SSTF	5	16
20	SCAN	5	16
21	C-SCAN	5	16
22	FSCAN	5	16
23	N-Step	5	16
24	Эшенбах	5	16
25	SSTF	3	8

4. Навыки, полученные студентом

После выполнения лабораторной работы студент должен:

– иметь представление о способах повышения эффективности работы дисковой подсистемы;

– знать основные стратегии оптимизации обработки запросов к дисковым накопителям;

– владеть методами выбора подходящей стратегии оптимизации;

– уметь на практике провести моделирование и объяснить его результаты.

Библиографический список

1. **Кнут Д. Е.** Искусство программирования. Т. 1. Основные алгоритмы. — 3-е изд. / Пер. с англ. : Учебное пособие — М.: Издательский дом «Вильямс», 2000. — 720 с.
2. **Фролов А. В., Фролов Г. В.** Аппаратное обеспечение IBM PC: В 2 ч. Ч. 1. — М.: «ДИАЛОГ-МИФИ», 1992. — 208 с. — (Библиотека системного программиста; Т. 2)
3. **Фролов А. В., Фролов Г. В.** Аппаратное обеспечение IBM PC: В 2 ч. Ч. 2. — М.: «ДИАЛОГ-МИФИ», 1992. — 208 с. — (Библиотека системного программиста; Т. 2)
4. **Дейтел Г.** Введение в операционные системы: В 2 т. Т. 2. / Пер. с англ. — М.: Мир, 1987. — 398 с.
5. **Столлингс В.** Операционные системы. — 4-е изд. / Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 848 с.

Оглавление

Учебное издание

Александр Викторович Чернышов

УПРАВЛЕНИЕ РЕСУРСАМИ ЭВМ

Редактор А. П. Головина

Оригинал-макет выполнен в пакете teTEX с использованием кириллических шрифтов семейства LH.

Вёрстка в TEXe: А. В. Чернышов

По тематическому плану внутривузовских изданий учебной литературы на 2004 г., поз. 77

Лицензия ЛР № 020718 от 02.02.1998 г.
Лицензия ПД № 00326 от 14.02.2000 г.

Подписано к печати	Формат 60×88/16
Бумага 80 г/см^2 «Снегурочка»	Ризография
Объем 2,0 п. л. Тираж 100 экз.	Зак. №

Издательство Московского государственного университета леса.
141005. Мытищи-5, Московская обл., 1-я Институтская, 1, МГУЛ.
Телефон: (095) 588-57-62
e-mail: izdat@mgul.ac.ru