Системное и прикладное программное обеспечение

Лабораторный практикум.
Методические указания к РГР.

Новосибирск 2016

Лабораторный практикум содержит методические указания к выполнению лабораторных работ и РГР по курсу «Системное и прикладное программное обеспечение».

Составитель: канд. физ.-мат. наук В. М. Неделько.

Работа подготовлена кафедрой ПС и БД.

Содержание.

Правила аттестации

В рамках курса требуется выполнить и защитить 7 лабораторных работ и РГР, а также сдать тест.

Лабораторные работы выполняются в бригадах по 1-2 человека. Численность бригады при выполнении РГР не ограничивается (однако, требуется указать роль и вклад каждого исполнителя).

Первые две работы из каждого из трёх разделов являются рекомендованными.

Оставшаяся работа выбирается на усмотрение учащегося из любых работ, в т.ч. из предлагавшихся в прошлые годы (27 работ).

Работы различаются по максимальному количеству баллов (весу).

Работы могут выполняться в среде Windows или Linux, при этом не менее 2-х работ должно быть выполнено под Linux.

Одна и та же работа не может выполняться и под Windows, и под Linux, за исключением работы по командной строке. Следует обратить внимание, что многие работы имеют отдельные описания для Windows и Linux, но эти варианты считаются одной работой.

Оценивание за семестр проводится по балльно-рейтинговой системе.

Лабораторные работы оцениваются в среднем в 5 баллов. Возможно начисление дополнительных баллов за оригинальное решение, существенно отличающееся от всех решений, уже показанных другими бригадами. Для некоторых работ дополнительные баллы ставятся бригаде, защитившей эту работу первой в группе (при условии оригинальности решения).

По двум работам требуется представить отчёт, который оценивается от 3-х до 5-ти, в отдельных случаях, до 7-ми баллов.

РГР оценивается до 20–25 баллов.

Тест состоит из 30 вопросов и оценивается до 10 баллов.

Устные ответы во время зачёта оцениваются до 10 баллов.

За обнаруженные неточности в описании лабораторных работ могут быть выставлены дополнительные баллы.

Зачёт выставляется при наборе не менее 50 баллов, при этом должна быть защищена РГР.

Оценка A+ не выставляется автоматически при достижении соответствующей суммы баллов, а ставится только при условии безупречного выполнения всех заданий.

Материал к каждой лабораторной работе разделен на четыре части:

— обзорная информация,

— вопросы для изучения,

— задание к выполнению,

— контрольные вопросы.

Защита работ предполагает демонстрацию выполненного задания, а также готовность практически продемонстрировать владение материалом и инструментарием, указанным в вопросах для изучения и дать ответы на контрольные вопросы.

Требования к отчету.

Отчёт оформляется по двум лабораторным работам на выбор, а также по РГР.

Все отчёты сдаются в печатном виде.

Отчет должен начинаться с заголовка, где указывается тема, исполнители и цель работы.

Основную часть составляет раздел «Ход работы», в котором должны быть описаны проделанные действия с объяснением результатов и обязательным приведением конкретных значений (имен файлов, путей, размеров архивов, номеров кластеров и т. п.). Порядок выполнения шагов может отличаться от рекомендованного в задании.

Вывод может содержать, например, заключение о степени удобства изученного программного продукта и возможной области его применения.

Раздел 1. Инструментарий пользователя ЭВМ

1.1. Командная строка

Лабораторная работа 1.1. Работа в режиме командной строки. Переменные окружения. Командные файлы.

Цель. Изучение принципов организации диалога пользователя с ОС средствами командной строки. Приобретение практических навыков по управлению ОС из командной строки. Ознакомление с операциями переадресации и фильтрации (шаблон имени файла). Получение начальных навыков по созданию пакетных файлов и использованию переменных окружения.

1.1.a. Вариант для ОС Windows

Вопросы для изучения.

1. Структура и основные элементы командной строки DOS.
2. Основные команды для манипулирования файлами.
3. Шаблон (маска) имени файла. Его использование.
4. Переменные окружения. Команда set.
5. Пакетные (командные) файлы.
   — Принцип организации.
   — Дополнительные возможности, по сравнению с командной строкой.
   — Ввод параметров. Использование переменных окружения.

Задание.

1. Запустив консоль (сеанс MS DOS, cmd.exe), выполните следующие действия.
   — Сделайте текущим каталог d:\work (команды d: и cd \work).
   — Создайте каталог my.
   — Скопируйте все файлы с расширением txt из work в my.
   — Выдайте содержимое my.
   — Измените системное приглашение (prompt).
   — Создайте текстовый файл (COPY CON file … Ctrl–Z).
   При необходимости пользуйтесь справкой.

Следующие действия можно выполнять из командной строки оболочки FAR–manager.

2. Используйте средства перенаправления вывода.
   — Создайте файл, содержащий список файлов текущего каталога с подкаталогами (требуется только список полных имён без другой информации).
   — Создайте файл со справкой для команды rar.
3. Выдайте значения переменных окружения. Выясните смысл нескольких переменных.
4. Создайте командные файлы с перечисленными ниже функциями.
   — Выдается сначала список файлов с расширением txt, затем с расширением doc.
   — Выдается содержимое каталога, указанного в переменной окружения TEMP (или TMP).
   — При наличии двух параметров производится копирование файла с именем, заданным первым параметром, во второй. Если параметр один, то файл выдается на терминал. При другом числе параметров выдается ошибка.
5. Найти в реестре раздел, описывающий созданную ассоциацию для расширения `.pas` (найти в `HKEY_CLASSES_ROOT` раздел `.pas`, запомнить значение по-умолчанию – это идентификатор типа, затем найти раздел с таким именем в корне `HKEY_CLASSES_ROOT`). Создать по аналогии раздел, связывающий .cpp с bc.exe (или с notepad.exe).

Контрольные вопросы.

1. Как сравнить два файла (по содержимому)?
2. Файлы с какими расширениями могут выполняться DOS?
3. Чем отличаются пути \my и my?
4. Для чего используется переменная окружения path?
5. Приведите примеры команд, которые работают в пакетных файлах, но не могут использоваться в командной строке.
6. Какие символы используются в маске имени файла?
7. Перечислите символы перенаправления ввода–вывода.
8. Что означают в DOS имена CON, PRN?
9. Для чего нужен реестр Windows?

1.1.b. Вариант для ОС Linux

Общие сведения.

Для взаимодействия с пользователем Unix-подобные системы поддерживают консоль и некоторое количество виртуальных или удаленных терминалов. После регистрации пользователя на терминале для него запускается интерпретатор команд, предоставляющий полный интерфейс к ОС.

В задачу командного процессора входит также интерпретация и выполнение командных файлов, предоставляя для их написания развитый скриптовый язык программирования.

Вопросы для изучения.

1. Структура и основные элементы командной строки. Специальные символы.
2. Основные команды для манипулирования файлами.
3. Операторы ‘;’, ‘&’, ‘&&’, ‘||’.
4. Перенаправление ввода-вывода (<,>,>>,|), каналы и фильтры.
5. Переменные окружения. Команда set.
6. Команда export.
7. Раскрытие выражений. Шаблон имени файла.
8. Командные файлы.
   — Принцип организации.
   — Условные операторы и циклы.
   — Ввод параметров.
   — Команда source или ‘.’ .
9. Команда su.

Задание.

1. Открыв консоль, выполните следующие действия.
   — Сделайте текущим домашний каталог (cd ~).
   — Создайте каталоги Work и Work/my.
   — Скопируйте все файлы с расширением txt из Work в my.
   — Выдайте содержимое my.
   — Измените системное приглашение (ps1), так чтобы оно стало похожим на приглашение DOS.
   — Создайте текстовый файл (cat >file … Ctrl–D).

   При необходимости пользуйтесь справкой.

   Следующие действия можно выполнять из командной строки оболочки mc.
2. Используйте средства перенаправления вывода.

— Создайте файл, содержащий список файлов текущего каталога с подкаталогами.
— Создайте файл со справкой для команды tar.

3. Выдайте значения переменных окружения. Выясните смысл нескольких переменных.
4. Создайте командные файлы с перечисленными ниже функциями.
   — Выдается сначала список файлов с расширением txt, затем с расширением doc.
   — Выдается содержимое каталога, указанного в переменной окружения TEMP (или TMP).
   — При наличии двух параметров производится копирование файла с именем, заданным первым параметром, во второй. Если параметр один, то файл выдается на терминал. При другом числе параметров выдается ошибка.

Контрольные вопросы.

1. Как сравнить два файла (по содержимому)?
2. Как отличить исполняемый файл?
3. Чем отличаются пути /my и my?
4. Для чего используется переменная окружения PATH?
5. Как передать вывод одной программы другой программе в качестве параметров?
6. Какие символы используются в маске имени файла?
7. Перечислите символы перенаправления ввода–вывода.
8. Как заставить команду cp вывести файл на терминал?

1.2. Издательская система TeX

Лабораторная работа 1.2. Издательская система TeX.

Цель. Освоение использования редактора TeX для оформления математических текстов.

Общие сведения.

Несмотря на предоставляемые в MS Word мощные средства по созданию документов, в ряде случаев этот продукт оказывается неудобным. В частности, Word малопригоден при работе с достаточно большими документами. В связи с этим достаточно актуальным может быть использование системы TeX, которая надежно работает с большими текстами и, кроме того, предоставляет значительно более удобный (при некотором навыке) инструмент по набору математических формул. Кроме того, TeX позволяет делать оформление более профессионального издательского уровня.

Вопросы для изучения.

1. Создание текстовых документов в LaTeX.
   — Заголовок и тело документа.
   — Форматирование текста.
   — Компиляция и просмотр.
2. Включение формул.
3. Обработка и включение графики.

Задание.

1. Оформите в TeX на выбор фрагмент (1 стр.) лекции по математическому анализу.
2. Возьмите из Интернет подходящую картинку. С помощью графического редактора преобразуйте ее в формат eps (для pdflatex нужен формат png), задав подходящие размеры, и вставьте в текст.
3. Создайте dvi и pdf файлы (утилиты …MiKTeX 2.5\miktex\bin\latex.exe и …pdflatex.exe). Оцените результат.
4. Скомпилируйте пример презентации с использованием пакета beamer.
5. Выберите любой математический журнал с требованиями представления работ в TeX. Оформите произвольный математический текст в соответствии с требованиями журнала.

Контрольные вопросы.

1. Чем отличаются математический и текстовый режимы?
2. Каковы особенности «выключенных» формул?
3. Назовите способы устранения ошибки **“Overfull hbox”**.
4. Как задать требуемый размер полей?
5. В чем отличие **\linebreak** от **\newline**?
6. Что содержит «шапка»?
7. Какие графические форматы поддерживает TeX?

1.3. Файловые менеджеры. Архивация данных

Лабораторная работа 1.3. Файловые процессоры. Архиваторы

Цель. Освоение возможностей файловых процессоров. Освоение типовых операций по работе с архивами. Изучение распространенных архиваторов.

1.3.а. Вариант для ОС Windows

Вопросы для изучения.

1. Операции с каталогами и файлами в FAR–manager.
   — Создание, переименование, перемещение, копирование, просмотр, удаление файлов и каталогов. Просмотр и изменение атрибутов файла. Просмотр информации об устройстве.
   — Поиск файлов и каталогов.
   — Связывание файлов с приложениями.
   — Подключение сетевых дисков.
2. Управление конфигурацией и режимы панелей.
   — Выделение файлов (files highlighting).
   — Сохранение конфигурации (Shift-F9).
   — Сортировка файлов на панели.
   — Сравнение каталогов.
   — Дерево каталогов.
   — Меню пользователя.
3. Редактирование в редакторе FAR.
   — Отличие кодировок DOS и Windows. Переключение русских и латинских букв в DOS и Windows приложениях.
   — Использование буфера обмена для копирования блоков текста внутри документа и для обмена текстом между документами: между двумя файлами в FAR; между FAR и Word.
4. Запуск приложений DOS.
   — Утилита start.
   — Запуск приложений из FAR в отдельной области памяти (Shift-Enter).
   — Настройка окна приложений DOS. Выбор шрифтов.
5. Работа с архиваторами из командной строки.
   — Типовые операции с архивом: создание, обновление, пополнение, удаление, просмотр оглавления.

— Параметры архивирования: сохранение структуры каталогов, архивация с паролем.
— Создание многотомных архивов.

6. Работа с архивами в FAR.
7. Архиватор WinZip.

Задание.

1. Открыть на панелях FAR два каталога с разных дисков (устройств). Выделить на первой панели (не подряд) несколько файлов (клавиша Ins), скопировать в личный каталог.
2. Связать *.pas с x:\bp\bin\bp.exe (или, например, с блокнотом) так, чтобы в окне редактирования появлялся открываемый файл (использовать !.! в качестве параметра команды).
3. Установить выделение каталогов и *.exe файлов.
4. Вставить произвольный текст из документа Word в текстовый файл в редакторе FAR. Добиться, чтобы в тексте одновременно присутствовали фрагменты в разных кодировках.
5. Открыть на панель FAR общедоступный каталог на ftp–сервере факультета.
6. Запустить другие файловые менеджеры. Сравнить с FAR, найти несколько наиболее существенных отличий.
7. Вызывая rar из командной строки, поместить заданные файлы в архив. Заархивировать файлы каталога соответствующие маске, задав дополнительно пароль архива. Извлечь файлы из архива.
8. С помощью rar создать многотомный архив, сохраняя структуру каталогов. Разархивировать файлы в другой каталог.
9. Работая в FAR, выделить заданные файлы и поместить их в архив. Выдать архив на панель (работа, как с каталогом). Извлечь заданные файлы из архива.
10. Создать архивный файл в WinZip.
11. Заархивировав один и тот же файл (достаточно большой) различными архиваторами, сравнить степени сжатия.

Контрольные вопросы.

1. Чем отличается связывание файлов с приложениями в FAR и в Windows?
2. По какому правилу выделяются файлы при выполнении сравнения каталогов?
3. Как перенести в командную строку имя текущего файла?
4. Как поместить в буфер обмена путь к текущему каталогу?
5. Чем отличаются кодировки DOS и Windows?

6. Как запустить из-под FAR приложение DOS в отдельной области памяти (в отдельном окне)?
7. Вы не помните имя файла, но знаете фрагмент содержимого, как можно найти требуемый файл?
8. Как можно результаты поиска сохранить в файле?
9. На чем основывается сжатие информации?
10. Что такое архивный файл?
11. В чем отличие команд *e* и *x* для rar?
12. Что такое многотомный архив?
13. Что означает самораспаковывающийся архив?
14. Что содержит оглавление архива?
15. Как осуществляется архивация с паролем?

1.3.b. Вариант для ОС Linux

Общие сведения.

Приведем основные определения.

Файл (file) – именованная совокупность данных, при этом полное имя файла (с включением пути) должно быть уникальным в пределах файловой системы. В Unix понятие файла используется также в более широком смысле, а именно, файлом является практически любой объект, с которым оперирует ОС.

Файловая система (ФС)– форма организации информации на носителе.

Виртуальная ФС – древовидная структура, включающая все файлы, с которыми оперирует ОС.

Монтирование ФС – присоединение устройства к виртуальной ФС. При монтировании устройство связывается с одним из существующих каталогов.

Каталог (директория, папка, directory, folder) – файл, содержащий список файлов с относящейся к ним служебной информацией, в формате, определенном файловой системой.

Устройство (том, device, volume) – носитель информации. Различают устройства *физические* – то, что фактически используется для хранения информации, и *логические* – то, чем накопитель информации представляется пользователю. Например, одно физическое устройство – жесткий диск может быть разделено на несколько частей – логических дисков, каждый из которых представляется пользователю отдельным устройством. Другой пример: сетевой диск – логическое устройство, физически соответствующее некоторому каталогу на диске другого компьютера.

Процесс или *задача* (process, task) – программа, запущенная на исполнение.

Вопросы для изучения.

1. Монтирование и размонтирование файловых систем.
2. Операции с каталогами и файлами в mc (Midnight commander).
   — Создание, переименование, перемещение, копирование, просмотр, удаление файлов и каталогов. Просмотр и изменение атрибутов файла. Просмотр информации об устройстве.
   — Поиск файлов и каталогов.
   — Связывание файлов с приложениями.
   — Подключение сетевых дисков.
3. Управление конфигурацией и режимы панелей.
   — Выделение файлов (files highlighting).
   — Сохранение конфигурации (Shift-F9).
   — Сортировка файлов на панели.
   — Сравнение каталогов.
   — Дерево каталогов.
   — Меню пользователя.
4. Редактирование во встроенном редакторе mc.
5. Запуск приложений.
6. Работа с архиваторами из командной строки.
   — Типовые операции с архивом: создание, обновление, пополнение, удаление, просмотр оглавления.
   — Параметры архивирования: сохранение структуры каталогов, архивация с паролем.
   — Создание многотомных архивов.
7. Работа с архивами в mc.
8. Архивация в KDE.

Задание.

1. На первую панель поместить любой доступный каталог с файлами небольшого размера, на вторую ~/work/my. Выделить на первой панели (не подряд) несколько файлов (клавиша Ins), скопировать в каталог ~/work/my.
2. Связать *.pas с freepascal (или с подходящим редактором) так, чтобы в окне редактирования появлялся открываемый файл.
3. Установить выделение каталогов и *.txt файлов.
4. Смонтировать ФС дискеты и скопировать на дискету несколько файлов.
5. Открыть на панель mc общедоступный каталог на ftp–сервере факультета.

6. Вызывая tar из командной строки, поместить заданные файлы в архив. Заархивировать файлы каталога соответствующие маске, задав дополнительно пароль архива. Извлечь файлы из архива.
7. С помощью tar создать многотомный архив, сохраняя структуру каталогов. Разархивировать файлы в другой каталог.
8. Сжать файл, используя gzip.
9. Работая в mc, выделить заданные файлы и поместить их в архив. Выдать архив на панель (работа, как с каталогом). Извлечь заданные файлы из архива.
10. Создать архивный файл в KDE.
11. Заархивировав один и тот же файл (достаточно большой) различными архиваторами, сравнить степени сжатия.

Контрольные вопросы.

1. Чем отличается связывание файлов с приложениями в mc и в Windows?
2. По какому правилу выделяются файлы при выполнении сравнения каталогов?
3. Как перенести в командную строку имя текущего файла?
4. Вы не помните имя файла, но знаете фрагмент содержимого, как можно найти требуемый файл?
5. Как можно результаты поиска сохранить в файле?
6. На чем основывается сжатие информации?
7. Что такое архивный файл?
8. Что такое многотомный архив?
9. Что означает самораспаковывающийся архив?
10. Что содержит оглавление архива?
11. Как осуществляется архивация с паролем?

1.4. Загрузчики. Эмуляторы

Лабораторная работа 1.4. Загрузчики. Эмуляторы.

Цель. Изучение процесса начальной загрузки. Знакомство с эмуляторами виртуальной машины и принципами одновременного запуска нескольких ОС.

Вопросы для изучения.

1. Формат загрузочного сектора.
2. Работа с неполными эмуляторами (VMWare, VirtualBox).
3. Настройка Bochs.
4. Запуск гостевой ОС.

Задание.

1. Настроить Bochs, подключив в качестве floppy disk A загрузочный образ MS DOS. Запустить эмуляцию и убедиться в работоспособности гостевой ОС.
2. Выполнить конфигурирование VMWare (или VirtualBox, или любой доступный эмулятор) для запуска Linux.
3. Настроить сеть в гостевой ОС.
4. Организовать обмен данными между ОС.

Контрольные вопросы.

1. Что такое эмулятор?
2. В чем отличие полных и неполных эмуляторов?
3. Может ли эмулятор запускать код для другой архитектуры?
4. В чем плюсы и минусы эмуляторов, исполняющих часть инструкций на реальном процессоре?

1.5. Программный интерфейс ОС

Лабораторная работа 1.5. Программный интерфейс MS Windows: Win API.

Цель. Изучение основ взаимодействия приложений с ОС Windows.

Все взаимодействие Windows с приложением основано на передаче сообщений. Каждое сообщение представляется 32-разрядным кодом.

Основным интерфейсным элементом Windows являются окна, поэтому именно окна выступают в качестве конечных адресатов сообщений. И действительно, большая часть сообщений логически привязано к определенному окну: перемещение, изменение размеров окна, щелчок мыши в области окна. Ввод с клавиатуры также привязывается к окну, являющемуся в данный момент активным.

Для реакции на сообщения каждому окну приложение должно сопоставить функцию обработки. Эта функция вызывается операционной системой автоматически для каждого пришедшего сообщения.

Однако первоначально сообщения попадают не в функцию окна-адресата, а в головной модуль приложения. Причем головной модуль windows–приложения обязан сам регулярно запрашивать сообщения, чтобы затем... отправлять их обратно операционной системе (обычно головной модуль содержит цикл, который вплоть до завершения программы выполняет две указанных операции). Такая идеология выглядит переусложненной, однако в ней есть определенная логика: функции окон играют роль обработчиков событий, а головной модуль имеет возможность контроля всех сообщений.

Ниже приводится пример минимальной программы для Windows.

Для понимания данного примера практически не требуется знание языка С, как такового, однако необходимо представление об основных элементах программ и общих принципах программирования.

```
#include <windows.h>

LRESULT CALLBACK WindowFunc(HWND, UINT, WPARAM, LPARAM);
char szWinName[]="MyWin";

                    // Головной модуль
int PASCAL WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst,
                   LPSTR lpszArgs, int nWinMode){
HWND hwnd,hwnd1;
```

```
MSG msg;
WNDCLASS wcl;

wcl.hInstance=hThisInst;
wcl.lpszClassName=szWinName;
wcl.lpfnWndProc=WindowFunc;
wcl.style=0;

wcl.hIcon=LoadIcon(NULL, IDI_APPLICATION);
wcl.hCursor=LoadCursor(NULL,IDC_ARROW);
wcl.lpszMenuName=NULL;

wcl.cbClsExtra=NULL; wcl.cbWndExtra=NULL;

wcl.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);
```

// Регистрация класса окна

```
if(!RegisterClass(&wcl)) return 0;
```

// Создание главного окна приложения

```
hwnd=CreateWindow(szWinName, "Window Skeleton",
     WS_OVERLAPPEDWINDOW,
     CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
     HWND_DESKTOP, NULL, hThisInst, NULL);
```

// Отрисовка окна

```
ShowWindow(hwnd, nWinMode);
UpdateWindow(hwnd);
```

// Выдача диалогового окна

```
MessageBox(hwnd,"Message","test",MB_OK);
```

// Цикл обработки сообщений

```
while(GetMessage(&msg,NULL,0,0)){
  TranslateMessage(&msg);
  DispatchMessage(&msg);}
return msg.wParam;
}
```

// Функция окна

```
LRESULT CALLBACK WindowFunc(HWND hwnd, UINT message,
                 WPARAM wParam, LPARAM lParam){
switch (message){
  case WM_DESTROY:
    PostQuitMessage(0); break;
```

```
    default:
      return DefWindowProc(hwnd,message,wParam,lParam);
    }
  return 0;}
```

Данная программа содержится в одном файле. Она может быть оттранслирована практически любым компилятором (например, Borland C 3.11).

Головной модуль программы сначала определяет и регистрирует в системе класс окна. Класс представляет собой совокупность свойств окна, в том числе, с ним связывается функция окна. Таким образом, два окна одного класса будут «близнецами».

Затем создается окно данного класса и вызывается функция его отображения.

Следующим шагом выдается окно сообщения (это, разумеется, необязательный момент, который добавлен лишь для «оживления» картинки).

Наконец, последний блок – это цикл обработки сообщений.

Функция окна сама обрабатывает лишь одно сообщение: **`WM_DESTROY`**, которое генерируется при закрытии окна (вызывается функция **`PostQuitMessage()`**, которая генерирует сообщение, завершающее приложение). Для остальных сообщений вызывается функция обработки по умолчанию: **`DefWindowProc()`**.

Вопросы для изучения.

1. Основные элементы взаимодействия приложения с ОС Windows.
   — Сообщения.
   — Цикл обработки сообщений.
   — Функция окна.
2. Этапы создания окна.
3. Элементы оконного интерфейса. Виды окон.

Задание.

1. Отредактировать и оттранслировать пример, приведенный выше.
2. Проверить работоспособность. Проследить выполнение шагов по тексту программы.
3. Модифицировать код, добавив создание второго окна, дочернего, с помощью вызова:

```
hwnd1=CreateWindow(szWinName, "Window 1",
     WS_OVERLAPPEDWINDOW | WS_CHILD ,
     CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
     hwnd, NULL, hThisInst, NULL);
```

Сравнить поведение окон.

4. Добавить выдачу дополнительного сообщения, например,
```
MessageBox(hwnd,"extra message","text",MB_YESNO);
```
5. Дополнительным плюсом может быть внесение собственных дополнений, например, обработки какого-либо сообщения, помимо **WM_DESTROY**.
6. Исследовать ассемблерный листинг кода системного вызова. Найти инструкции, непосредственно относящиеся к вызову.

Контрольные вопросы.

1. Какие типы окон Вы знаете?
2. Какую особенность имеют модальные окна диалога (сообщения)?
3. В чем особенность дочерних окон?
4. Для чего нужен цикл сообщений?

Раздел 2. Управление ресурсами ВС

2.1. Файловая система (FAT, ext2fs)

Лабораторная работа 2.1. Файловая система.

Цель. Ознакомление с файловой системой. Приобретение навыков по восстановлению файловой структуры средствами низкого уровня.

2.1.а. Вариант для ОС Windows

Вопросы для изучения.

1. Физическая модель диска. Физический адрес сектора.
2. Логическая модель диска. Основные элементы файловой системы.
3. Загрузочная запись.
4. Организация таблицы размещения файлов.
5. Структура каталога.
6. Интерфейс программы diskeditor.

Задание.

Работа выполняется под эмулятором в виртуальной DOS.

1. Скопировать на виртуальную дискету произвольный текстовый файл, размером не менее 2-х кб. Удалить его. Восстановить файл в diskedit-е "вручную", т. е. непосредственно корректируя запись в каталоге (dir) и таблицу размещения файлов (FAT).
2. Скопировать на дискету два текстовых файла. Объединить их в один, не перемещая данные, а корректируя цепочку кластеров в FAT и информацию в dir.
3. Создать в diskedit новый текстовый файл (выбрать свободный кластер, записать в него текст, создать запись в каталоге и цепочку в FAT).
4. Освоить операцию поиска в diskedit (найти кластер с записанным текстом).
5. Проверить дискету NDD или ScanDisk. Исправить ошибки, если таковые допущены.

В отчете описать последовательность действий (указывая конкретные размеры файлов, в какие элементы FAT какие числа записываются и т. д.).

Контрольные вопросы.

1. В каком кластере находится корневой каталог?
2. Перечислите элементы записи каталога.
3. Каков принцип хранения длинных имен файлов в каталоге системы FAT?
4. Чем отличается физический и логический адрес сектора?
5. Каков размер сектора? Кластера?
6. Какие записи являются обязательными для некорневого каталога?
7. Что содержит загрузочный сектор?

2.1.b. Вариант для ОС Linux

Общие сведения.

Сокращение ext2fs расшифровывается как "вторая расширенная файловая система" (second extended filesystem). При этом имеются в виду расширения по отношению к ФС minix, использованной в первых версиях Linux.

Рассматриваемая файловая система состоит из следующих элементов:

— начальный (загрузочный) блок,
— суперблок,
— область индексных дескрипторов (INOD-s),
— обрасть данных.

Весь объем логического диска разбит на блоки фиксированного размера, обычно по 1024 байта. Первый блок является резервным и служит для хранения загрузчика, если такой имеется.

Суперблок содержит общую информацию о ФС, в частности размеры ее элементов.

Область индексных дескрипторов является массивом записей, описывающих файлы. Каждая запись (INOD) хранит следующую информацию о файле:

— тип файла,
— время создания и модификации,
— идентификаторы владельца и группы владельца,
— права доступа и атрибуты,
— номера блоков.

В большинстве файловых систем поддерживается семь типов файлов

— обычные файлы,
— каталоги,
— файлы байт-ориентированных (символьных) устройств,
— файлы блок-ориентированных (блочных) устройств,
— сокеты,

— именованные каналы (FIFO),

— символические ссылки.

В данной работе нас будут интересовать только обычные файлы и каталоги. Каталог представляет собой файл, представляющий собой список файлов и состоящий из массива записей, включающих имя файла и номер индексного дескриптора. Заметим, что один и тот же файл может быть описан в разных каталогах (в том числе под разными именами).

В INOD хранится информация о размещении файла на диске. Имеется массив из 13 элементов, первые 10 из которых являются непосредственно номерами блоков, занятых файлом, 11-й элемент указывает на блок 1-го уровня косвенности, 12-й и 13-й — на блоки 2-го и 3-го уровней соответственно.

Блок 1-го уровня содержит продолжение массива блоков, занятых файлом (если файл имеет размер более 10 блоков). Если одного блока 1-го уровня недостаточно для описания всех блоков, занятых файлом, то используется блок 2-го уровня, содержащий массив номеров дополнительных блоков первого уровня. аналогичным образом при необходимости используется блок 3-го уровня.

Учитывая, что один блок может содержать до 256 указателей на блоки, получаем, что максимально допустимый размер файла в ext2fs составляет: $1024 \cdot (10 + 256 + 256 \cdot 256 + 256 \cdot 256 \cdot 256) = 17,247,250,432$ байт.

Для выполнения работы необходим доступ на запись к устройству с ext2fs, это может быть виртуальный диск или FDD. В последнем случае дискету необходимо отформатировать командой: mkfs -t ext2 /dev/fd0

Также для работы потребуется дисковый редактор. Это может быть, например lde (Linux Disk Editor) и утилита debugfs из пакета e2fsprogs.

Вопросы для изучения.

1. Физическая модель диска. Физический адрес сектора.
2. Логическая модель диска. Основные элементы файловой системы.
3. Загрузочная запись (резервный блок).
4. Суперблок.
5. Структура INOD.
6. Структура каталога.
7. Интерфейс дискового редактора.

Задание.

Работа выполняется под эмулятором в виртуальном Linux.

1. Скопировать на дискету (виртуальный диск) произвольный текстовый файл, размером не менее 2-х кб. Удалить его. Восстановить файл, пользу-

ясь дисковым редактором, непосредственно корректируя запись в каталоге и соответствующий INOD.

2. Скопировать на дискету два текстовых файла. Объединить их в один, не перемещая данные, а корректируя цепочку блоков в индексных дескрипторах и информацию в каталогах.
3. Создать новый текстовый файл (выбрать свободный блок на диске, записать в него текст, создать запись в каталоге и INOD).
4. Освоить операцию поиска (найти блок с записанным текстом).
5. Проверить целостность ФС. Исправить ошибки, если таковые допущены.

В отчете описать последовательность действий (указывая конкретные размеры файлов, содержимое INOD и т. д.).

Контрольные вопросы.

1. В каком блоке находится корневой каталог?
2. Перечислите элементы записи каталога.
3. Каков принцип хранения информации о блоках, занятых файлом?
4. Чем отличается физический и логический адрес сектора?
5. Каков размер сектора? Блока?
6. Какие записи являются обязательными для каталога?
7. Что содержит загрузочный сектор?
8. Что содержит суперблок?
9. Как хранится информация о свободных блоках?
10. Как отличить свободный INOD ?

2.2. Производительность функций чтения и записи файлов

Лабораторная работа 2.2. Измерение производительности функций работы с файлами.

Цель. Выработка навыков эффективной работы с файлами программными средствами.

Общие сведения.

Для выполнения работы можно взять за основу следующий код.

```
#include <fstream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/timeb.h>

static int delta_time(const _timeb &t1, const _timeb &t2)
{
  return (int) ((t2.time - t1.time) * 1000
            + ((int) t2.millitm) - ((int) t1.millitm));
}

using namespace std;

static int test_fprintf()
{
  _timeb time0;
  _timeb time1;

  FILE* t = fopen("test_fprintf.txt","wt");

  _ftime(&time0);

  for (int i=0; i<500000; i++)
    fprintf(t,"%i %i %i %i\n",i*4,i*4+1,i*4+2,i*4+3);

  _ftime(&time1);

  fclose(t);
  return delta_time(time0,time1);
}

static int test_ofstream()
{
  _timeb time0;
  _timeb time1;

  ofstream outfile("test_ofstream.txt");

  _ftime(&time0);

  for (int i=0; i<100000; i++)
    outfile << i*4 << ' '<< i*4+1<< ' ' << i*4+2<< ' ' << i*4+3 << '\n';
```

```
  _ftime(&time1);

  outfile.close();
  return delta_time(time0,time1);
}

static int test_ifstream()
{
  _timeb time0;
  _timeb time1;

  ifstream infile("test_ofstream.txt");

  _ftime(&time0);

  int x1,x2,x3,x4;

  for (int i=0; i<100000; i++){
    infile >> x1 >> x2 >> x3 >> x4;
    //printf("%i %i %i %i\n",x1,x2,x3,x4);
    }

  _ftime(&time1);

  infile.close();
  return delta_time(time0,time1);
}

static int test_fscanf()
{
  _timeb time0;
  _timeb time1;

  FILE* t = fopen("test_fprintf.txt","rt");

  _ftime(&time0);

  int x1,x2,x3,x4;

  for (int i=0; i<500000; i++){
    fscanf(t,"%i %i %i %i\n",&x1,&x2,&x3,&x4);
    //printf("%i %i %i %i\n",x1,x2,x3,x4);
    }

  _ftime(&time1);

  fclose(t);
  return delta_time(time0,time1);
}

static long filesize(FILE *stream)
{
  const long curpos = ftell(stream);
  fseek(stream, 0L, SEEK_END);
  const long length = ftell(stream);
  fseek(stream, curpos, SEEK_SET);
  return length;
}

static int test_fread()
{
```

```
  _timeb time0;
  _timeb time1;

  FILE* b = fopen("test_fprintf.txt","rb");
  const long size = filesize(b);

  char* buf = new char[size];

  _ftime(&time0);

  const int N = (int) fread(buf,1,size,b);

  _ftime(&time1);

  fclose(b);
  delete [] buf;
  return delta_time(time0,time1);
}

int main(int argc, char* argv[])
{
  const int dt_fprintf = test_fprintf();
  printf( "\ntest_fprintf takes %i milliseconds\n", dt_fprintf);

  const int dt_ofstream = test_ofstream();
  printf( "\ntest_ofstream takes %i milliseconds\n", dt_ofstream);

  const int dt_ifstream = test_ifstream();
  printf( "\ntest_ifstream takes %i milliseconds\n", dt_ifstream);

  const int dt_fscanf = test_fscanf();
  printf( "\ntest_fscanf takes %i milliseconds\n", dt_fscanf);

  const int dt_fread = test_fread();
  printf( "\ntest_fread takes %i milliseconds\n", dt_fread);

  getch();
  return 0;
}
```

Вопросы для изучения.

1. Библиотека stdio.h
2. Библиотека iostream.
3. Поэлементное и буферизованное чтение.
4. Работа с текстовыми файлами.
5. Фрагментация файлов.
6. Кэширование.

Задание.

1. Сравните быстродействие fprintf и <<.
2. Сравните быстродействие >> и комбинации fgets+sscanf.

3. Сравните быстродействие поэлементного и буферизованного вывода. Проследите зависимость от размера буфера.
4. Оцените эффект кэширования.
5. Сравните скорость чтения фрагментированных и нефрагментированных файлов. Задание не является обязательным, за выполнение даются дополнительные баллы.

Контрольные вопросы.

1. Чем iostream лучше stdio.h?
2. Чем stdio.h лучше iostream?
3. Сформулируйте основные рекомендации по увеличению скорости работы с файлами.

2.3. Производительность функций управления памятью

Лабораторная работа 2.3. Измерение производительности функций управления памятью.

Цель. Выработка навыков эффективной работы с динамической памятью.

Вопросы для изучения.

1. Классы памяти: сегмент данных, стек, куча.
2. Функции выделения и освобождения памяти.
3. Операторы new и delete.
4. Фрагментация кучи.
5. Файл подкачки.

Задание.

1. Сравните быстродействие при выделении памяти при различных размерах блоков.
2. Сравните быстродействие при последовательном и хаотическом чередовании выделения и освобождения памяти, а также со случаем, когда размер блока случаен.
3. Сравните быстродействие при выделении стековой памяти и памяти в куче.
4. Оцените эффект использования файла подкачки.

Контрольные вопросы.

1. Сформулируйте основные рекомендации по увеличению быстродействия манипулирования памятью.

Раздел 3. Программные средства для математических рассчётов

Лабораторная работа 3.1. Матричные вычисления в Matlab.

Цель. Освоение базовых средств Matlab для работы с матрицами.

Вопросы для изучения.

1. Основные элементы рабочего пространства Matlab.
2. Ввод значений матриц и векторов.
3. Основные операции над матрицами.
4. Сохранение данных в бинарных файлах.
5. Импорт данных из текстовых файлов, вывод в текстовые файлы.

Задание.

1. Выполните следующие вычисления.
   - — Пользуясь непосредственным вводом значений, задайте матрицу A, размером 3 на 3, и вектор свободных коэффициентов b.
   - — Вызвав соответствующую функцию, получите решение системы линейных уравнений $Ax = b$.
   - — Убедитесь в правильности решения путём подстановки, вычислите невязку (меру точности решения).
   - — Найдите матрицу, обратную к A.
   - — Убедитесь в том, что $x = A^{-1}b$.
   - — Найдите жорданову форму матрицы A.
   - — Получите сингулярное разложение матрицы A.
   - — Вычислите A^{-1}, пользуясь сингулярным разложением матрицы A.
2. Рабочая область.
   - — Сохраните рабочую область, перезапустите Matlab и восстановите рабочую область.
   - — Выдайте информацию о рабочей области (о переменных) в кратком и в подробном виде.
   - — Очистите рабочую область.
3. Освойте использование файлов для хранения данных.
   - — В Excel задайте матрицу, размером 5 на 5, сохраните файл в формате csv.
   - — Импортируйте файл в Matlab.
   - — Найдите обратную матрицу, сохраните её в файл в формате csv.

— Откройте файл в Excel, средствами Excel убедитесь в правильности нахождения обратной матрицы.
— Продемонстрируйте в Matlab сохранение матрицы в бинарный файл и чтение из этого файла.

Контрольные вопросы.

1. Чем различаются вектор-строка и вектор-столбец, как их задать?
2. Что входит в рабочее пространство?

Лабораторная работа 3.2. Средства визуализации в Matlab.

Цель. Освоение базовых средств визуализации Matlab.

Вопросы для изучения.

1. Файлы кода в Matlab.
2. Двумерные графики.
3. Трёхмерные графики.

Задание.

1. Выполните следующие вычисления.
   — Пользуясь непосредственным вводом значений, задайте некоторую симметричную матрицу A, размером 4 на 4.
   — Найдите жорданову форму матрицы A. Убедитесь в ортогональности матрицы перехода.
   — Получите сингулярное разложение матрицы A. Убедитесь, что жорданова форма совпадает с сингулярным разложением.
   — Создайте m-файл с кодом, выполняющим перечисленные выше действия. Убедитесь в его работоспособности.
2. Двумерные графики.
   — Постройте график некоторого полинома третьей степени.
   — Найдите корни полинома.
   — Постройте в одном окне два графика.
   — Постройте спираль, пользуясь параметрическим заданием функции в декартовых координатах.
   — Постройте график в полярных координатах.
3. Трёхмерные графики.
   — Пользуясь непосредственным вводом значений, задайте некоторую симметричную матрицу A, размером 2 на 2.

— Постройте график квадратичной формы y = x’Ax.
— Приведите матрицу A к диагональному виду. Постройте график квадратичной формы с диагональной матрицей.
— Постройте графики двумерной плотности вероятности, представляющей собой смесь двух нормальных распределений. В качестве альтернативы можно взять любую функцию двух переменных с неединственным локальным экстремумом.
— Постройте контурную диаграмму.
— Постройте график пространственной кривой.

Контрольные вопросы.

1. Для чего используется функция meshgrid?

Лабораторная работа 3.3. Программирование в Матлаб. Взаимодействие Матлаб с приложениями на C++

Цель. Получение навыков программирования в Matlab. Изучение средств взаимодействия Matlab с кодом на C++.

Работа оценивается с весом 1,5–2,0.

Вопросы для изучения.

1. Создание новых функций.
2. Подключение C++ модулей в Матлаб.
3. Вызов Матлаб из программ на C++.

Задание.

1. Создать функцию для вычисления площади эллипса, задаваемого уравнением 1 = x’Ax.
2. Реализовать эту же функцию на C++ и осуществить её вызов из среды Матлаб. Убедиться в правильности полученного результата.
3. Вызывая Матлаб из программы на C++, построить график одномерной функции по своему выбору.

Лабораторная работа 3.4. Математические расчёты в R

Цель. Получение навыков работы в среде R.

Работа оценивается с весом 1,5–2,5, при условии оригинальности решения.

Вопросы для изучения.

1. Особенности синтаксиса R в сравнении с Матлаб.
2. Средства статистического анализа данных.

Задание.

1.a. На портале kaggle.com выбрать любую из учебных задач. Запустить на выбор один из скриптов, предлагаемых для её решения. Добиться совпадения полученных результатов с опубликованными.

1.b. Альтернативное задание: выполнить в R все задания работы 3.1.

Расчётно-графическая работа

Тема работы: проектирование и разработка программного обеспечения.

Цель работы: сформировать практические навыки проектирования и разработки программных продуктов.

Требования к выполнению

Работа выполняется в рамках коллективной (желательно) или индивидуальной разработки программного продукта в соответствии с вариантом задания.

Объем работ: 15 часов.

Допускается использование готовых свободно-распространяемых кодов.

Этапы выполнения

Этапы работы:

— составление проектной документации,
— написание кода,
— отладка и тестирование,
— инсталляция, составление пользовательской документации.

Данные задачи выполняются индивидуально или распределяются между членами бригады. Задачи достаточно независимы. В частности, набор тестов составляется по спецификации и не требует наличия кода. Руководство пользователя также можно написать на основе лишь проекта, не имея реализации.

Структура отчета

1) Аннотация проекта (описание идеи и цели).
2) Введение в предметную область (или реферат).
3) Спецификация (техническое задание на разработку).
4) Архитектура, технические решения (основной раздел).
5) Примеры кода с описанием.
6) Набор тестов.
7) Презентация продукта (описание, что получилось).
8) Руководство пользователя.

Отчет может быть представлен один на бригаду или персонально.

Обязательно использование UML.