ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного
бюджетного образовательного учреждения высшего образования
«Московский технический университет связи и информатики»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ
для проведения практических занятий по дисциплине «Распределенные операционные системы» ФТД.01

Кафедра **«Информатика и вычислительная техника»**

Направление подготовки **09.03.01. Информатика и вычислительная техника**

Профиль **Программное обеспечение и интеллектуальные системы**

Формы обучения **очная, заочная**

Разработала:
Доцент кафедры ИВТ Швидченко С.А.

Содержание

ПРАКТИЧЕСКАЯ РАБОТА № 1.

Изучение основ работы в Лазарус. Вставка надписей и графики. Создание кнопок и программирование переходов. Вставка звука и видео. Вставка текста

1. Компоненты среды Delphi и Lazarus. Репозиторий объектов и эксперты.

Обращение к справочной системе Help

Для вызова справочной системы необходимо выбрать соответствующую команду в выпадающем меню Help или отметить элемент интерфейса в исходном тексте и нажать клавишу F1. При нажатии кнопки “Разделы” в окне Help появляется диалоговое окно справочной системы Windows 95, которое позволяет вам просмотреть содержание всех файлов Help группы, отыскать ключевое слово по индексу или начать процесс поиска.

Меню и команды Delphi

Чтобы выдать команду в среде Delphi, можно воспользоваться тремя основными способами:

- С помощью меню.
- С помощью полоски SpeedBar (инструментальной линейки).
- С помощью SpeedMenu (одного из локальных меню, которое активизируется при нажатии правой кнопки мыши).

Меню File

Команды выпадающего меню File можно использовать для работы, как с проектами, так и с файлами исходного кода.

К командам, работающим с проектами, относятся New, New Application, Open, Reopen, Save Project As, Save All, Close All, Add to Project и Remove from Project. С файлами исходного кода работают команды New, New Form, New Data Module, Open, Reopen, Save As, Save, Close и Print. Основной командой является File/New, которую можно использовать для вызова экспертов, для начала работы с новым приложением, для наследования формы из уже существующей и т.д. Чтобы открыть проект или файл исходного кода, с которыми вы работали последний раз, используйте команду File/Reopen.

Меню Edit

Стандартные возможности меню Edit применимы как к тексту, так и к компонентам формы. Можно копировать и вставлять тот или иной текст в редакторе, копировать и вставлять компоненты в одной форме или из одной формы в другую. Также можно копировать и вставлять компоненты в другое групповое окно той же формы, например, в панель или блок группы; копировать компоненты из формы в редактор, и наоборот. Delphi помещает компоненты в буфер обмена, преобразуя их в текстовое описание. Можно соответствующим образом отредактировать этот текст, а затем вставить его обратно в форму в виде нового компонента. Можно выбрать несколько компонентов и скопировать их как в другую форму, так и в текстовый редактор. Это может пригодиться, когда вам придется работать с рядом схожих компонентов. Вы сможете скопировать один компонент в редактор, размножить его нужное число раз, а затем вставить назад в форму целую группу.

Меню Search

Если вы выберете команду Incremental Search, то вместо того чтобы показать диалоговое окно, где вводится образец для поиска, Delphi переходит в редактор. Когда вы введете первую букву, редактор перейдет к первому слову, которое начинается с этой буквы. Продолжайте набор букв и, курсор будет последовательно переходить к словам, в начале которых будут стоять введенные символы. Эта команда очень эффективна и чрезвычайно быстра. Команда Browse Symbol вызывает Object Browser – инструмент, который можно использовать для просмотра многих деталей при исследовании откомпилированной программы.

Меню View

Большинство команд меню View применяются для отображения какого-либо окна среды Delphi, например Project Manager, Breakpoints List или Components List. Эти окна не связаны друг с другом. Команда Toggle Form/Unit

используется для перехода от формы, над которой вы работаете к ее исходному коду, и обратно. Команда New edit window создает дубликат окна редактирования и его содержимого. В Delphi это единственный способ просмотреть два файла рядом друг с другом, поскольку редактор для показа нескольких загруженных файлов использует ярлычки. После дублирования окна редактирования могут содержать разные файлы. Последние две команды меню View можно использовать для удаления с экрана полоски SpeedBar и палитры Components, хотя при этом среда Delphi становится менее удобнее для пользователя. Команда Build All заставляет Delphi откомпилировать каждый исходный файл проекта, даже если после последней трансляции он не был изменен. Для проверки написанного кода без создания программы можно использовать команду Syntax Check. Команда Information дает некоторые подробности о последней выполненной вами трансляции. Команда Options применяется для установки опций проекта: опций компилятора и редактора связей, опций объекта приложения и т.д.

Меню Run

Меню Run можно было бы назвать Debug (отладка). Большинство команд в нем относится к отладке, включая саму команду Run. Программа, запускаемая внутри среды Delphi, выполняется в ее интегрированном отладчике (если не отключена соответствующая опция). Для быстрого запуска приложения используется клавиша F9. Остальные команды применяются в процессе отладки для пошагового выполнения программы, установки точек прерывания, просмотра значений переменных и объектов, и т.п.

Меню Component

Команды меню Component можно использовать для написания компонентов, добавления их в библиотеку, а также для конфигурирования библиотеки или палитры компонентов.

Меню Tools

Меню Tools содержит список нескольких внешних программ и инструментальных средств. Команда Tools позволяет сконфигурировать это выпадающее меню и добавить в него новые внешние средства. Меню Tools также включает команду для настройки репозитория и команду Options, которая конфигурирует всю среду разработки Delphi.

Полоска кнопок быстрого доступа SpeedBar

Наиболее часто используемые команды Delphi имеются в инструментальной линейке SpeedBar. Изменить размеры SpeedBar можно буксируя толстую линию между ней и палитрой Ccomponents. Другие разрешенные в SpeedBar операции добавляют, удаляют или заменяют пиктограммы с помощью команды Configure собственного локального меню SpeedBar. Эта операция вызывает инструмент SpeedBar Editor. Чтобы добавить пиктограмму в SpeedBar, необходимо найти ее в нужной категории и отбуксировать в полоску. Подобным образом можно отбуксировать пиктограмму за пределы SpeedBar или просто передвинуть ее в другое место.

Локальные меню. SpeedMenu

Хотя меню Delphi содержит большое количество элементов, не все команды доступны через выпадающие меню. Иногда для некоторых окон или областей окна приходится использовать SpeedMenu (локальное меню). Чтобы активизировать SpeedMenu, нужно нажать над необходимым элементом интерфейса пользователя правую кнопку мыши или клавиши Alt и F10.

Работа с формами

Проектирование форм – ядро визуальной разработки в среде Delphi. Каждый помещаемый в форму компонент или любое задаваемое свойство сохраняется в файле, описывающем форму (DFM-файл), а также оказывает некоторое влияние на исходный текст, связанный с формой (PAS-файл).

Можно начать новый пустой проект, создав пустую форму или начать с существующей формы (используя различные доступные шаблоны) или добавить в проект новые формы. Проект (приложение) может иметь любое число форм.

При работе с формой можно обрабатывать ее свойства, свойства одного из ее компонентов или нескольких компонентов одновременно. Чтобы выбрать форму или компонент, можно просто щелкнуть по нему мышью или воспользоваться Object Selector (комбинированный список в Object Inspector), где всегда отображены имя и тип выбранного элемента. Для выбора нескольких компонентов можно или нажать клавишу Shift и щелкать по компонентам левой кнопкой мыши, или отбуксировать в форме рамку выбора.

SpeedMenu формы содержит ряд полезных команд. Для изменения относительного расположения компонентов одного вида можно использовать команды Bring to Front и Send To Back. Командой Revert To Inherited можно воспользоваться, чтобы в унаследованной форме установить те значения свойств выбранного компонента, которые были у них в родительской форме. При выборе сразу нескольких компонентов вы можете выровнять их или изменить их размеры.

С помощью SpeedMenu можно также открыть два диалоговых окна, в которых устанавливается порядок обхода визуальных управляющих элементов и порядок создания невизуальных управляющих элементов. Команда Add To Repository добавляет текущую форму в список форм, доступных для использования в других проектах.

Для установки положения компонента кроме применения мыши имеются еще два способа:

- Установка значений для свойств Top и Left.
- Использование клавиш курсора при нажатой клавише Ctrl.

Метод Ctrl+клавиша курсора особенно удобен при тонкой подстройке положения элемента. Точно также, нажимая клавиши курсора при нажатой клавише Shift, можно подстроить размер компонента.

Палитра компонентов

Чтобы добавить в текущую форму новый компонент, можно щелкнуть на одной из страниц палитры Components, а затем, чтобы разместить новый элемент, щелкнуть в форме. Причем в форме можно или буксировать мышь с нажатой левой кнопкой, чтобы установить сразу и размер, и положение компонента, или просто щелкнуть один раз, позволяя Delphi установить размер по умолчанию.

Каждая страница палитры содержит ряд компонентов, которые обозначены пиктограммами и именами, появляющимися ввиле подсказки. Эти имена являются официальными названиями компонентов. В действительности это названия классов, описывающих компоненты без первой буквы T (например, если класс называется Tbutton, имя будет Button). Если необходимо поместить в форму несколько компонентов обного и того же вида, то при выборе компонента щелчком в палитре удерживайте нажатой клавишу Shift Затем при каждом щелчке в форме Delphi будет вставлять новый компонент выбранного вида.Чтобы остановить эту операцию, просто щелкните по стандартному селектору (пиктограмма стрелки) слева от палитры Components.

Object Inspector

Object Inspector используется при проектировании формы для установки свойств компонента (или самой формы). В его окне в двух колонках изменяемого размера перечислены свойства (или события) выбранного элемента и их значения. Окно Object Selector в верхней части Object Inspector указывает текущий компонент и его тип данных, и этот селектор можно использовать для изменения текущего выбора.

В Object Inspector перечислены не все свойства компонентов, а только те, что могут быть установлены на этапе проектирования. Правая колонка Object Inspector разрешает правильное редактирование в соответствии с типом данных свойства. В зависимости от свойства можно вставить строку или число, выбрать из списка опций (на эту возможность указывает стрелка) или вызвать специальный редактор (об этом говорит овальная кнопка). Для некоторых свойств, таких, как Color, разрешен и ввод значения, и выбор элемента из списка, и вызов специального редактора.

Написание кода

При проектировании формы в Delphi обычно приходится писать кое-какой код для отклика на некоторые из ее событий. Когда вы нажимаете кнопку мыши в форме или на компоненте, Windows посылает вашему приложению сообщение, информируя его об этом событии. Реакция Delphi состоит в получении сообщения о событии и вызове соответствующего метода отклика на событие. Для каждого вида компонентов Delphi определяет ряд различных событий. Вы можете узнать о событиях, доступных для формы или компонента, рассматривая страницу Events окна Object Inspector в тот момент, когда выбран необходимый элемент.

Вставим в форму компонент – кнопку – и заставим его откликаться на событие, связанное с нажатием левой кнопки мыши. При щелчке по кнопке мышью в работающей программе возникает событие OnClick (По щелчку). Пока это событие никак не обрабатывается программой и поэтому нажатие кнопки не приведет ни к каким последствиям. Чтобы заставить программу реагировать на нажатие кнопки, необходимо написать на языке Object Pascal фрагмент программы, который называется обработчиком события. Фрагмент оформляется в виде специальной подпрогаммы – процедуры.

Чтобы заставить Delphi самостоятельно сделать заготовку для процедуры обработчика события OnClick, дважды щелкните мышью по вновь вставленному компонету. В ответ Delphi активизирует окно кода и вы увидите в нем такой текстовый фрагмент:

```
procedure  TForm1.Button1Click(Sender: Tobject);
begin

end;
```

Слово **procedure** извещает компилятор о начале подпрограммы –процедуры. За ним следует имя процедуры TForm1.Button1Click. Это имя – составное: оно состоит из имени класса TForm1 и собственно имени процедуры Button1Click.

Каждый компонент принадлежит к строго определенному классу, а все конкретные экземпляры компонентов, вставляемые в форму, получают имя класса с добавленным числовым индексом. По используемому в Delphi соглашению все имена классов начинаются с буквы T. Таким образом, имя TForm1 означает имя класса, созданного по образцу стандартного класса Tform. Если вы посмотрите начало текста в окне кода, то увидите следующие строки:

```
type
   TForm1 = class(TForm)
      Button1: Tbutton;
      procedure  TForm1.Button1Click(Sender: Tobject);
  private
      {private declarations}
  public
      {public declaration}
  end;
var
   Form1: TForm1;
```

Строка TForm1 = **class**(TForm) определяет новый класс TForm1, который порожден от стандартного класса TForm. Стандартный класс TForm описывает пустое Windows-окно, в то время как класс TForm1 описывает окно с уже вставленным в него компонентом *кнопка.*

Каждый раз когда вы работаете над событием, Delphi открывает редактор с исходным файлом, связанным с формой. Редактор позволяет работать над несколькими файлами исходного текста одновременно, используя образ “записной книжки с ярлычками”. Каждая страница записной книжки соответствует отдельному файлу.

Компиляция проекта

Компиляция проекта происходит при запуске его на выполнение. Транслируя проект, Delphi компилирует только те файлы, которые изменялись. Но если вы выберите в меню Project команду Build All, то будет откомпилирован каждый файл, даже если он не изменялся.

Проект содержит список файлов исходного кода, которые являются частями проекта, а также соответствующих им форм (при наличии таковых). Сначала каждый файл исходного кода превращается в откомпилированный модуль Delphi – файл с тем же именем, что и у исходного файла на языке Паскаль, но с расширением DCU.

Во время компиляции и компоновки самого проекта, откомпилированные модули, составляющие проект, сливаются (или связываются) друг с другом и с кодом библиотеки VCL, образуя исполнимый файл.

Интегрированный отладчик

В Delphi имеется встроенный отладчик, обладающий огромным количеством возможностей. При каждом запуске из среды Delphi, программа уже выполняется в отладчике. Для установки точки останова или щелкните в промежутке между левой рамкой окна редактирования и текстом, или выберите в SpeedMenu команду Toggle Breakpoint, или нажмите клавишу F5. Когда вы разместили ряд точек останова, можете использовать команду Breakpoints меню View, чтобы открыть окно Breakpoints List. Один из пунктов в верхней части окна Breakpoints List предполагает добавление условия в точке останова, так чтобы программа выполнялась только при выполнении данного условия. Кнопка Step Over на линейке SpeedBar позволяет просмотреть выполнение операторов один за другим, а кнопка Trace Into позволяет трассировать вызываемые методы (т.е. выполнять шаг за шагом код подпрограмм).

Если программа остановлена в отладчике, вы можете проверить значение любого идентификатора (для переменных, объектов, компонентов, свойств и т.д.), который доступен в этой точке программы. Для этого существуют два способа: использовать диалоговую панель Evaluate/Modify или добавить элемент в окно Watch List. Самый простой способ открыть диалоговую панель Evaluate/Modify – выделить переменную в редакторе исходного текста, а затем выбрать команду Evaluate/Modify из SpeedMenu редактора. Вы можете устанавливать элементы наблюдения, используя команду Add Wath at Cursor в Speed Menu редактора.

Файлы, создаваемые системой

Когда вы сохраняете новый проект, Delphi создает ряд файлов. Здесь приводится списо наиболее важных файлов.

- Основной файл проекта типа .DPR. Это основной модуль исходного текста проекта. Имеется только один DPR-файл для каждого проекта. Этот файл, кроме всего прочего, перечисляет имена других файлов, составляющих проект.
- Файлы формы типа .DFM. Это двоичные файлы ресурсов, содержащие определение визуальных форм. В проекте может быть много форм и каждая имеет собственный .DFM файл.
- Файл модуля Паскаля типа .PAS. Содержит код Object Pascal для соответствующей формы или для автономного модуля кода.
- Файл опций проекта типа .OPT. Файл, который содержит различные установки Delphi (текстовый файл).
- Откомпилированные файлы модуля типа .DCU. Содержат объектный код существующего .PAS-файла модуля.
- Откомпилированные программные файлы типа .EXE. Это собственно программы Windows.
- Откомпилированные файлы динамических библиотек типа .DLL. Это откомпилированные модули Windows, которые могут использоваться одновременно многими программами Windows.

Страницы репозитория объектов

В Delphi есть несколько команд меню, с помощью которых вы можете создать новую форму, новое приложение, новый модуль данных, новый компонент и т.п. Эти команды находятся в меню File, а также в других выпадающих меню. Но если вместо них выдать команду File/New, Delphi откроет окно Object Repository.

Репозиторий используется для создания новых элементов любого вида: форм, приложений, модулей данных, библиотек, компонентов и т.д. Диалоговое окно Object Repository содержит несколько страниц:

- Страница New позволяет создавать новые элементы многих разных типов.
- Страница текущего проекта (в действительности на ярлычке данной страницы вы увидите имя проекта, например Project1) позволяет унаследовать форму или модуль данных от аналогичного объекта, включенного в ваш текущий проект.
- Страницы Forms, Dialogs , Data Modules позволяют создавать новые формы, диалоговые панели и модули данных, используя эксперты или существующие объекты этих типов.
- Страница Project позволяет скопировать файлы из хранящегося в репозитории проекта.

В нижней части диалогового окна Object Repository находятся три радиокнопки, с помощью которых можно указать: хотите ли вы скопировать существующий элемент, унаследовать его или применить непосредственно, не копируя.

Страница New

Список элементов, которые можно создать с помощью этой страницы:

- Application создает новый пустой проект.
- Data Module создает новый пустой модуль данных.
- DLL создает новую библиотеку DLL.
- Form создает новую пустую форму.
- Text открывает в редакторе новый текстовый файл.
- Unit создает новый пустой модуль, не связанный с формой.

Страница Forms

Ниже приведен список необходимых для работы предопределенных форм:

- About Box – простая панель “О программе”.
- Duil List Box – форма с двумя разными списками; позволяет пользователю выбрать ряд элементов в одном списке и нажатием кнопки переместить их во второй. Кроме компонентов эта форма содержит значительное количество не очень простого кода на языке Паскаль.

Страница Dialogs

Эта страница похожа на предыдущую, но содержит другие элементы.

- Dialog Expert – простой эксперт, который способен сгенерировать различные диалоговые панели с одной или несколькими страницами.
- Dialog with help – два варианта диалоговой панели с кнопкой вызова справочной информации.
- Password dialog – диалоговая панель с простым окном редактирования, которая имеет необходимые для ввода пароля опции; код отсутствует.
- Standart Dialog Box – стандартная диалоговая панель, которая доступна в двух вариантах с различным расположением кнопок.

Страница Data Modules

Модуль данных это особый вид формы, который никогда не появляется на экране во время выполнения и может использоваться для хранения невизуальных компонентов. Чаще всего он применяется для описания доступа к базе данных.

Страница Projects

Эта страница содержит схемы проектов, которые вы можете использовать в качестве стартовой площадки для создания собственного приложения.

- Application Expert – простой эксперт, в котором вы можете выбрать файловую структуру и некоторые другие элементы приложения.
- MDI Application задает ключевые элементы программы с интерфейсом Multiple Document Interface (MDI). В этом приложении определена основная форма для окна MDI-фрейма, содержащая меню, строку состояния и инструментальную линейку. Кроме того, в нем имеется вторая форма, которую на этапе выполнения можно использовать для создания дочерних окон.
- SDI Application определяет основную форму со стандартными атрибутами современного интерфейса пользователя, включая инструментальную линейку и строку состояния, а также типичную панель About.

- Win95 LogoApplication описывает простое приложение, в котором присутствует большинство элементов, необходимых для получения логотипа Windows 95. Данная команда в основном создает SDI-приложение с компонетом RichEdit и вставляет в него код, который делает приложение совместимым с электронной почтой.

Эксперты Delphi

Delphi разрешает не только копировать или использовать существующий код, но и создавать новые формы, приложения или другие файлы с кодом, применяя эксперт. Эксперт позволяет вам ввести ряд опций и с помощью некоторой внутренней схемы создает код, соответствующий вашему заказу.

Application Expert

Его можно активизировать из страницы Project окна Object Repository. Первая страница этого эксперта позволяет добавить в программу некоторые стандартные выпадающие меню: File, Edit, Window и Help. На второй странице эксперта вы зададите расширения тех файлов, которые должна рассматривать программа. Вам придется ввести как описание файла, например Текстовый файл (*.txt) так и его расширение txt Эти величины будут использоваться в качестве значений по умолчанию диалоговыми окнами File Open и File Save, которые Application Expert добавит в программу (если вы выбрали поддержку файлов).

Application Expert выведет на экран прекрасное визуальное средство, которым вы можете воспользоваться для построения инструментальной линейки. В нем вы выбираете одно из выпадающих меню и появляется ряд стандартных кнопок, которые соответствуют его типичным элементам (но только, если это меню было выбрано на первой странице эксперта).

После завершения работы над инструментальной линейкой вы можете перейти на последнюю страницу эксперта. Здесь устанавливаются многие дополнительные опции, например, можно доказать поддержку интерфейса MDI, добавить строку состояния или разрешить всплывающие подсказки. Вы также можете задать имя нового приложения и указать каталог для его исходных файлов. Каталог для приложений должен уже существовать. Если вы хотите поместить файлы проекта в новый каталог, выберите кнопку Browse и введите новый путь – появится диалоговое окно с вопросом, хотите ли вы создать новый каталог.

Dialog Box Expert

Это простой эксперт, предоставленный вместе с исходным текстом в качестве демонстрационного примера. Вы можете воспользоваться этим экспертом как инструментом для построения диалоговых панелей двух различных видов: простых и многостраничных диалоговых панелей. Если вы выберите простую диалоговую панель, эксперт перейдет к третьей странице, где вы сможете задать компоновку кнопок. Если вы выберите многостраничную панель, появится промежуточная страница, которая позволяет ввести тексты для разлимчных ярлычков.

Задание № 1.1

- Изучить вышеизложенный материал с помощью компьютера.
- Создать с помощью Application Expert приложение, содержащее выпадающие меню File, Edit, Window и Help, инструментальную линейку, строку состояния и всплывающие подсказки. Приложение записать в каталог C:/Program Files/Borland/Delphi3/user. Если такой каталог не существует, создать его. Исследовать в окне редактирования полученный код.
- Построить с помощью Dialog Box Expert простую и многостраничную диалоговые панели.
- Создать MDI Application, SDI Application, Win95 LogoApplication. Исследовать код. Приложение на диске не запоминать.

2. Типы данных и операторы языка Паскаль.

Тип определяет значения, которые может иметь переменная, и операции, которые могут быть выполнены над этой переменной. После слова var сопровождается список имен переменных, сопровождаемых двоеточием и именем типа данных.

Предопределенные типы данных

Существует несколько предопределенных типов данных, которые можно разделить на три группы: перечислимые типы, вещественные типы и строки.

Перечислимые типы

Три наиболее важных перечислимых типа –Integer (целочисленный), Boolean (логический) и Char (символьный). Однако существует несколько других типов, которые имеют тот же смысл, но иное внутреннее представление и диапазон значений.

Ниже приведен полный список перечислимых типов:

- Integer, Cardinal, ShortInt, SmallInt, LongInt, Byte, Word
- Boolean, ByteBool, WordBool, LongBool
- Char, ANSIChar, WideChar

Задание № 2.1

Сконструировать форму с шестью кнопками, имена которых ShortInt, SmallInt, Integer, Byte, Word, Cardinal; с четырьмя статическими надписями (компонент Label) Type, Size, Max, Min и четырьмя надписями для вывода информации о типе при каждом нажатии одной из кнопок. Для этого записать для каждой кнопки метод отклика на событие OnClick, используя свойство Caption надписей для вывода информации и функции SizeOf – размер внутреннего представления переменной данного типа, High – самое высокое значение в диапазоне перечислимого типа, Low – самое низкое значение, а также функцию IntToStr – преобразование числа в строку. Пример строки кода:

```
SizeLabel.Caption:=IntToStr(SizeOf(Number));
```

Вещественные типы

Вещественные типы представляют разнообразные форматы чисел с плавающей запятой. Меньше всего памяти требуется доля хранения чисел типа Single. Затем идут числа типа real, Double и Extended. Все это – типы чисел с плавающей запятой, имеющие разную точность представления.

Кроме них есть еще тип данных Comp, который описывает очень длинные целые числа, и Currency – тип данных, который имеет четыре десятичных знака после запятой и 64-битовое внутреннее представление. Последний тип данных был добавлен для показа больших денежных сумм без потери младших значащих цифр. Вещественные ипы используются в программах, содержащих математические формулы. Сама Delphi использует вещественные типы в типе данных TdateTime.

Типы данных, специфичные для Windows

В Delphi имеются типы данных, которые определены системой Windows – дескриптор и ссылка на цвет. Их имена, соответственно, Thandle и TcolorRef. Первый тип – это лишь переопределение типа данных Cardinal, а второй переопределение типа LongInt.

Ссылка на цвет – это просто число, описывающее цвет. Вы можете выбрать любой цвет с помощью функции RGB или напрямую значение для красной, зеленой и синей составляющих величины типа TcolorRef.

В Windows дескриптор – это число, которое является ссылкой на внутреннюю структуру данных системы. Например, когда вы работаете с окном, система выдает вам дискриптор окна (HWND). Тем самым она сообщает, что окно, с которым вы имеете дело, является окном под номером, например, 142. С этого момента ваше приложение может использовать данный номер, чтобы попросить систему обработать необходимое окно: переместить его, изменить размеры, уменьшить до пиктограммы и т.п.

Другими словами, дескриптор является внутренним кодом, который вы можете применить для обращения к конкретному элементу, обрабатываемому системой, включая окна, растровые изображения, пиктограммы, блоки памяти, курсоры, шрифты, меню и т.п.

Приведение и преобразование типов

Вы не можете присвоить переменной значение другого типа. Если в этом все же возникла необходимость, имеются две возможности. Первая возможность – приведение типов, которое выглядит как простой вызов функции, но вмето имени функции используется имя типа данных адресата:

```
var
   N:Integer;
   C:Char;
   B:Boolean;
begin
    N:= Integer('X');
    C:= Char(N) ;
    B:= Boolean(0);
```

Строго говоря, операцию приведения можно осуществлять между теми типами данных, которые имеют одинаковый размер. Обычно безопасным является приведение между перечислимыми или между вещественными типами, но вы также можете выполнить приведение между типами указателей (а также объектов).

Вторая возможность – использовать подпрограмму преобразования типов, например – Trunc – преобразует значение вещественного типа в значение целочисленного типа, отсекая дробную часть; IntToStr – преобразует число в строку; StrToInt – преобразует строку в число, вызывая исключение в случае неправильной строки и т.д.

Массивы

Используйте для работы с массивами функции Low и High (особенно в циклах), поскольку они делают код независимым от диапазона массива. Если позже вы измените объявленный диапазон индексов массива, то код, который использует Low и High, останется работоспособным, а код, который жестко привязан к диапазону массива работать не будет. Функции Low и High облегчают поддержку вашего кода и делают его более надежным. Применение этих функций не приводит к лишним затратам на этапе выполнения. Во время компиляции они преобразуются в константные выражения, а не вдействительные обращения к функциям.

Длинные строки

Чтобы устранить ограничения традиционных строк Паскаля, в Delphi введена поддержка длинных строк. В действительности имеется два типа строк:

- Тип ShortString соответствует обычным строкам Паскаля. Каждый элемент короткой строки имеет тип ANSIChar.
- Тип AnsiString соответствует новым длинным строкам переменного размера. Такие строки размещаются динамически и их размер практически не ограничен. В основе таких строк также лежит тип ANSIChar.

В зависимости от значения новой директивы компилятора $H вы получите или короткую, или длинную строку. По умолчанию стоит значение $H+, что соответствует длинным строкам.

Операторы языка Паскаль

Здесь будет дано краткое описание использования операторов Object Pascal в Delphi.

Рассмотрим пример, который демонстрирует различие между фиксированным счетчиком и циклом с псевдослучайным счетчиком. Начните новый пустой проект и поместите в его основную форму список и две кнопки. Теперь в событие OnClick кнопок можно добавить некоторый код. Первая кнопка содержит простой цикл for для отображения списка чисел. До выполнения этого цикла, который добавляет несколько строк в свойство Items списка, вы должны очистить содержимое самого списка.

```
procedure TForm1.Botton1.Click (Sender:TObject);
   var
     I:Integer;

begin
   ListBox1.Items.Clear;
    For I:= 1 to 20 do
        ListBox1.Items.Add(' String ' + IntToStr(I));
end;
```

Код, связанный со второй кнопкой использует цикл while, который основан на счетчике, увеличивающемся случайным образом.

```
procedure TForm1.Botton2.Click(Sender:TObject) ;
   var
       I:Integer;

begin
   ListBox1.Items.Clear;
   Randomize;
   I:=0;
   while  I<1000 do
       begin
           I:=I+Random(100);
           ListBox1.Items.Add(' Random number: ' + IntToStr(I));
       end;
end;
```

При каждом щелчке по второй кнопке числа будут различными, так как они зависят от генератора случайных чисел.

Элементы управления редактированием

Класс TCustomEdit – это абстрактный класс для всех элементов управления редактированием в Delphi. Он включает простой элемент управления редактированием, элементы управления редактированием по маске и все элементы управления мемо.

Некоторые свойства и методы, реализованные классом TCustomEdit

Используйте или установите это…	***Чтобы сделать это …***

Brush	Определить цвет и шаблон, используемые в качестве фона оконного элемента управления
CanFocus	Определить, может ли оконный элемент управления получить фокус
Clear	Очистить содержимое элемента управления редактированием
Enabled	Определить доступность элемента управления
Focused	Определить находится ли оконный элемент управления в фокусе
Font	Определить шрифт, используемый для вывода текста в элементе управления
GetSelTextBuf	Скопировать выбранный текст из элемента управления в буфер
GetTextBuf	Скопировать текст из элемента управления в буфер
GetTextLen	Получить длину текста элемента управления
Hide	Сделать элемент управления невидимым
Hint	Определить текст, который отображается в подсказке для элемента управления
SelectAll	Выбрать весь текст в элементе управления
SelLength	Определить длину выбранного текста в элементе управления
SelStart	Определить исходную позицию выбранного текста
SelText	Получить доступ к выбранному тексту в элементе управления редактирования
SetFocus	Установить фокус на оконный элемент управления
Show	Сделать элемент управления видимым
Text	Обратиться к изменяемому тексту на элементе управления

Класс TEdit инкапсулирует большинство возможностей стандартного элемента управления редактированием известного как “поле” или “ текстовое поле”. Элемент управления редактированием предоставляет одну доступную для редактирования строку текста внутри элемента управления с необязательной рамкой. При желании текст на элементе управления редактированием может быть предназначен только для чтения, так что пользователь изменять его не сможет.

Класс TEdit предусматривает только основные функциональные возможности элемента управления редактированием. При необходимости ограничить диапазон ввода, воспринимаемый этим элементом управления, используйте вместо него элемент управления редактированием по маске (TMaskEdit). Класс TEdit порожден непосредственно от TCustomEdit.

Задание № 2.2

Сконструировать форму, которая будет содержать следующие управляющие элементы:

- Окно редактирования со связанной с ним меткой Operand 1. В этом окне вводится первый операнд.
- Окно редактирования со связанной с ним меткой Operator. В этом окне вводится операция. В программе предусмотрены операции +, -, / и * .
- Окно редактирования со связанной с ним меткой Operand 2. В этом окне вводится второй операнд.
- Окно редактирования со связанной с ним меткой Result. В этом окне отображается результат запрошенной вами операции.
- Кнопка Close, которая закрывает приложение.

Тип операндов – целый.

Задание № 2.3

Сконструировать форму, которая будет содержать следующие управляющие элементы:

- Элемент управления редактированием по маске (TMaskEdit) со связанной с ним меткой Operand 1.
- Элемент управления редактированием по маске (TMaskEdit) со связанной с ним меткой Operand 2.
- Элемент управления редактированием по маске (TMaskEdit) со связанной с ним меткой Result.
- Кнопка Close, которая закрывает приложение.
- Шесть кнопок операций + , - , * , / , mod , div .
- Кнопка Clear для очистки окон элементов управления редактированием.

Тип операндов – вещественный. Написать код, который будет реализовывать указанные операции по нажатию соответствующей кнопки (связать код с событием OnClick кнопки).

3. Классы и модули.

Классы и сокрытие информации

Класс может содержать сколько угодно данных и любое количество методов. Однако для соблюдения всех правил объектно-ориентированного подхода данные должны быть скрыты, или инкапсулированы, внутри использующего их класса. Использование метода для получения доступа к внутреннему представлению объекта уменьшает риск возникновения ошибочных ситуаций и позволяет автору класса модифицировать внутреннее представление в будущих версиях. В Object Pascal имеются две различные конструкции, подразумевающих инкапсуляцию, защиту и доступ к переменным: классы и модули. С классами связаны некоторые специальные ключевые слова – спецификаторы доступа:

- private – элементы интерфейса класса видны только в пределах модуля, в котором определен класс. Вне этого модуля private-элементы не видны и недоступны. Если в одном модуле создано несколько классов, то все они видят private-разделы друг друга.
- Раздел public не накладывает ограничений на область видимости перечисленных в нем полей, методов и свойств. Их можно вызывать в любом другом модуле программы.
- Раздел published также не ограничивает область видимости, однако в нем перечислены свойства, которые должны быть доступны не только на этапе исполнения, но и на этапе конструирования программы. Раздел published используется при разработке компонентов. Без объявления раздел считается объявленным как published. Такой умалчиваемый раздел располагается в самом начале объявления класса любой формы и продолжается до первого объявленного раздела. В раздел published среда помещает описание вставляемых в форму компонентов. Сюда не нужно помещать собственные элементы или удалять элементы, вставленные средой.
- Раздел protected доступен только методам самого класса, а также любым потомкам, независимо от того, находятся ли они втом же модуле или нет.

Порядок следования разделов может быть любой, любой раздел может быть пустым.

Классы и модули

Приложения Delphi интенсивно используют модули. За каждой формой скрывается соответствующий ей модуль. Однако модули не обязаны иметь соответсвующие формы.

Модуль содержит раздел interface, где объявлено все, что доступно для других модулей, и раздел implementation с реальным кодом. Наконец, модуль может иметь два необязательных раздела : initialization с некоторым кодом запуска, который выполняется при загрузке в память программы, использующей данный модуль, и finalization, который выполняется при завершении программы.

Предложение uses в начале раздела interface к каким другим модулям мы должны получить доступ из раздела interface текущего модуля. Если же на другие модули необходимо сослаться из клда подпрограмм и методов, вы должны добавить новое предложение uses в начале раздела implementation.

В интерфейсе модуля можно объявить несколько различных элементов, в том числе процедуры, функции, глобальные переменные и типы данных. Также можно поместить в модуль класс. Delphi это делает автоматически при создании каждой формы. Чтобы создать новый не относящийся к форме модуль, выберите команду File/New и отметьте на странице New появившегося окна Object Repository элемент Unit.

Модули и область видимости

Область видимости идентификатора (переменной, процедуры, функции или типа данных, определяет ту часть кода, в которой доступен этот идентификатор. Основное правило состоит в том, что идентификатор является значимым только внутри его области видимости.

Если вы объявили идентификатор внутри блока определения процедуры, вы не сможете использовать данную переменную вне этой процедуры. Область видимости идентификатора охватывает всю процедуру, включая вложенные блоки.

Если вы объявили идентификатор в области реализации модуля, вы не можете применить его вне модуля, номожете использовать в любом блоке и процедуре, определенных внутри модуля. Если вы объявили идентификатор в интерфейсной части модуля, его область
видимости распространяется на любой другой модуль, где объявлен идентификатор. Любой
идентификатор, объявленный в интерфейсе модуля, является глобальным; все другие идентификаторы принято называть локальными.

Модули и программы

Приложение Delphi создается из файлов исходного текста двух разных видов. Это один или несколько модулей и один файл программы.

Модули можно считать вторичными файлами, к которым обращается основная часть приложения – программа. На практике файл программы обычно является автоматически сгенерированным файлом с ограниченной ролью. Он нужен только для запуска программы, которая выполныет основную форму. Код файла программы, или

файла проекта Delphi (DPR), можно отредактировать вручную или с помощью Project Manager и некоторых опций проекта.

Информация о типе на этапе выполнения

Правило языка Object Pascal о совместимости типов для классов-потомков позволяет вам использовать класс-потомок там, где ожидается класс предок, обратное невозможно. Теперь предположим, что класс Dog содержит функцию Eat, которая отсутствует в классе Animal.

Если переменная MyAnimal ссылается на объект типа Dog, вызов этой функции должен быть разрешен. Но если вы попытаетесь вызвать эту функцию, а переменная ссылается на объект другого класса, возникнет ошибка. Поскольку компилятор не способен определить, будет ли значение правильным на этапе выполнения, то делая явное приведение типов, мы рискуем вызвать опасную ошибку этапа выполнения программы.

Для решения данной проблемы можно воспользоваться некоторыми подходами, основанными на системе RTTI. Каждый объект знает свой тип и своего предка и можем получить эту информацию с помощью операции is. Параметрами операции is являются объект и тип:

if MyAnimal is Dog then …

Выражение is становится истинным, только если в настоящее время объект MyAnimal имеет тип Dog или тип потомка от Dog. Другими словами, это выражение приобретает значение True, если вы можете без риска присвоить объект (MyAnimal) переменной заданного типа данных (Dog). Такое прямое приведение можно выполнить так:

```
if MyAnimal is Dog then
   MyDog:= Dog(MyAnimal) ;
```

То же действие можно выполнить напрямую с помощью другой операции RTTI – as. Мы можем написать так:

```
MyDog:= MyAnimal as Dog ;
Text:= MyDog.Eat ;
```

Если мы хотим вызвать функцию Eat, можно использовать и другую нотацию:

```
(MyAnimal as Dog).Eat ;
```

Результатом выражения будет объект с типом данных класса Dog, поэтому выможете применить к нему любой метод этого класса.

Приведение с операцией as отличается от традиционного приведения тем, что в случае несовместимости типа объекта с типом, к которому вы пытаетесь его привести, порождается исключение EInvalidCast.

Чтобы избежать этого исключения, используйте операцию is и в случае успеха делайте простое приведение:

```
if MyAnimal is Dog then
  (Dog(MyAnimal)).Eat;
```

Задание № 3.1

Открыть модуль, не связанный с формой и поместить в него три класса:

- Класс Animal, который содержит в разделе public объявление конструктора Create и объявление метода-функции: Verse – звук, издаваемый животным. Тип результата возвращаемого функцией – string. Метод Verse объявить виртуальным и абстрактным. В разделе private класса определить переменную Kind : string.
- Класс Dog объявить потомком класса Animal. В разделе public этого класса объявить конструктор и методы Verse и Eat. Метод Eat типа string объявить виртуальным (пища животного).
- Класс Cat объявить потомком класса Animal. Раздел public класса содержит те же определения, что и соответствующий раздел класса Dog.

В реализациях конструктора каждого класса переменной Kind присваивается имя соответствующего животного, напрамир для класса Animal : Kind := ‘An Animal’.

В реализациях методов Verse возвращается звук, издаваемый животным, например Verse := ‘Mieow’.

В реализациях методов Eat возвращается название пищи, которой питается соответствующее животное.

- Задать имя модуля и имя проекта, в который этот модуль будет включен.
- Добавить в проект форму, которой присвоить имя Animals, также задать имя модулю, связанному с формой.
- В форме расположить три кнопки опций (компонент RadioButton) с названиями Animal, Dog, Cat ; кнопкой команды (компонент Button) с названием Kind и две крупный надписи (компонент Label) . Нажатию одной из кнопок опций будет соответствовать выбор животного. При нажатии кнопки команды надписи должны отобразить звук, издаваемый животным и его пищу.
- Определите в классе формы private-переменную MyAnimal. Запишите код для обработчика события OnCreate формы, где создается объект типа Dog на который ссылается переменная MyAnimal.
- В обработчиках события OnClick каждой кнопки опций записать код, который удаляет текущий объект и создает новый.
- В обработчике события OnClick кнопки команды записать код, который будет помещать в надписи звук, издаваемый животным и его пищу. Для работы с методом Eat используйте операцию is для приведения типов.
- Если вы все сделали правильно, при запуске приложения надписи будут отображать пищу и звук для Dog и Cat и приложение завершит работу по ошибке при выборе Animal.

- Уберите ключевое слово abstract в объявлении метода Verse. Запустите приложение снова. Посмотрите что изменилось в работе приложения. Объясните различия.
- Попробуйте использовать метод Eat без приведения типов (без is).
- Разработайте два класса потомка от Dog, которые будут отображать особенности двух пород собак. Разработайте методы для этих классов, позволяющие получить некоторые характеристики породы (рост, длина шерсти, длина ушей и т.д.). Дополните форму компонентами позволяющими увидеть все характеристики разработанных классов.

4. Использование компонентов

Буксировка из одного компонента в другой

В Delphi буксировка обычко выполняется путем нажатия кнопки мыши над одним компонентом и освобождения над другим. Для этой операции можно написать код, который обычно копирует в компонент адресат свойство, значение или что-нибудь еще.

Задание № 4.1

- Создать форму, в которой расположить четыре надписи с названиями каждого цвета и надпись-адресат с некоторым описательным текстом. Цвет фона надписей задается с помощью свойства Color. Цвет текста можно задать используя свойство Font: на этапе проектирования открывается окно. Для того чтобы разрешить буксировку: выбрать значение dmAutomatic для свойства DragMode для надписей источников и предоставить методы отклика для пары событий в надписи-адресате.
- Первое событие – OnDragOver. Оно вызывается, когда во время буксировки вы перемещаете курсор над компонентом. Это событие указывает, что компонент воспринимает буксировку. Обычно событие считается завершенным после определения следующего факта: имеет ли компонент Source (тот, который инициализировал операцию буксировки) необходимый тип. Для события OnDragOver записать следующий код :

```
Accept:= Source is TLabel;
```

Этот код воспринимает операцию буксировки, активизируя соответствующий курсор, если только исходный объект действительно является объектом надписи.

- Второй метод, который вы должны написать, соответствует событию OnDragDrop:

```
Label5.Color:= (Source as TLabel).Color;
```

Обработка исключений

На этапе выполнения Delphi порождает исключения, когда какой-либо процесс идет неправильно. Если код вашей подпрограммы написан соответствующим образом, он может распознать возникшую проблему и попытаться ее решить; в противном случае исключение передается в код, который вызвал вашу подпрограмму и т.д. В конечном счете, если никто не обработал исключение, его обрабатывает Delphi, выводя на экран стандартное сообщение об ошибке и пытаясь продолжить выполнение программы.

Весь механизм строится на четырех ключевых словах:

- ✓ try – определяет начало защищенного блока кода;
- ✓ except – определяет конец защищеннго блока кода и вводит операторы обработки исключений в следующем виде: on (тип исключения) do (оператор)
- ✓ finally – указывает необязательный блок, который используется для освобождения ресурсов, распределенных в блоке try перед обработкой исключения; этот блок завершается ключевым словом end.
- ✓ raise – оператор, используемый для порождения исключений. Большинство исключений, которые вы встретите при программировании на Delphi, будут порождаться системой, но вы также можете создать их в собственном коде, когда во время выполнения обнаружатся недопустимые или несогласованные данные. Кроме того, ключевое слово raise можно использовать внутри обработчика для повторного порождения исключения, т.е. для передачи его следующему обработчику.

Примеры стандартных классов исключений:

- EAbort – реализует обработку любого исключения без сообщения;
- EConvertError – Ошибка преобразования в функциях StrToInt или StrToFloat;
- EDivByZero – ошибка целочисленного деления на ноль.

Если вы хотите, чтобы при правильной обработке исключений программа продолжала выполняться, отключите опцию отладки Break on Exception в окне Environment Options.

Восприятие ввода для пользователя

Обратим особое внимание на качество, характерное для многих управляющих элементов – фокус. Как определить какое окно или поле имеет фокус ввода? В каждый конкретный момент фокус имеет только одно поле. Вы можете перемещать фокус, используя клавишу Tab или щелкая мышью по другому компоненту. Каждый раз когда компонент получает или теряет фокус, к нему приходит соответствующее событие, которое указывает, что пользователь достиг (OnEnter) или покинул (OnExit) компонент.

При использовании компонента Edit для ввода чисел пользователь вместо цифры может набрать букву. Функции преоразования вернут код ошибки, что поможет определить действительно ли введено число. Когда можно выполнить такую проверку? Возможно, когда изменится значение блока редактирования, когда компонент потеряет фокус или когда пользователь щелкнет по некоторой кнопке в диалоговой панели. Можно просматривать входной поток в блоке редактирования и останавливать любой нечисловой код.

Задание № 4.2

- Создать форму с пятью полями редактирования и пятью соответствующими надписями, которые поясняют, какой вид проверки осуществляет соответствующий компонент Edit. Форма также содержит кнопку для проверки содержимого первого поля редактирования.
- Записать следующий код для события OnClick кнопки:

```
var
  Number,Code:Integer ;
begin
   if Edit1.Text<>' '
     then begin
            val( Edit1.Text,Number,Code);
            if Code <> 0
              then begin
                     Edit1.SetFocus;
                     MessageDlg(' Not a number in the first edit ',mtError,[mbOK],0);
                   end;
         end;
end;
```

- Для события OnExit компонента Edit2 записать следующий код

```
var
  Number,Code:Integer;
begin
  if (Sender as TEdit).Text <> ' '
   then begin
          val((Sender as TEdit).Text,Number,Code);
          if Code <> 0
           then begin
                  (Sender as TEdit).SetFocus;
                 MessageDlg('The  edit  field  number  '  +  IntToStr((Sender  as
TEdit).Tag)
                  + 'does not have a valid number',mtError,[mbOK],0) ;
                end;
       end;
end;
```

Этот код можно использовать для любого компонента TEdit. Текст сообщения об ошибке можете написать свой.

- Третий компонент Edit выполняет аналогичную проверку при каждом изменении его содержимого (используя событие OnChange}.
- Компонент Edit имеет событие OnKeyPress, которое соответствует нажатию клавиши пользователем. Записать код для этого события компонента Edit4:

```
begin
   if not (key in['0'..'9',#8]}
     then begin
            Key:= #0;
            MessageBeep($FFFFFFFF);
          end;
end;
```

- Для события OnEnter компонента Edit5 записать код, в котором необходимо преобразовать введенные символы в число с помощью функции StrToInt. Использовать исключение для обработки ошибки EConvertError.

5. Использование компонентов

Создание редактора для текста формата RTF

Windows 95 содержит новый управляющий элемент, который способен поддерживать формат Rich Text Format (RTF). Компонент Delphi Rich Edit инкапсулирует поведение этого стандартного управляющего элемента. Поместите в форму три компонента: Panel, которая занимает верхнюю часть формы; Button, расположенная на панели; RichEdit, который занимает всю остальную часть формы. Нажатие на кнопку должно изменять шрифт выделенного фрагмента текста в окне RichEdit. Программа отображает стандартное диалоговое окно Font, использовав в качестве начального значения исходный шрифт компонента RichEdit. Затем выбранный пользователем шрифт скопируется в атрибуты текущего выбранного текста. Хотя свойства DefAttributes и SelAttributes компонента RichEdit не имеют тип TFont, они совместимы с ним, поэтому для копирования значения можно применить метод Assign:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if RichEdit1.SelLength>0 then
  begin
    FontDialog1.Font.Assign(RichEdit1.DefAttributes);
     if FontDialog1.Execute then
       RichEdit1.SelAttributes.Assign(FontDialog1.Font);
  end
  else
    ShowMessage ('No text selected');
end;
```

Задание № 5.1

Создать вышеописанную форму и записать код для события OnClick кнопки. В форму поместить компонент FontDialog.

Выбор

Существуют два стандартных управляющих элементаWindows, которые позволяют пользователю выбирать разные опции. Первый – чекбокс. Он соответствует опции, которую можно выбрать независимо от других. Второй управляющий элемент – кнопка опций, которая соответствует исключительному выбору. Например, если вы видите две кнопки опций с метками A и B, вы можете выбрать или A, или B, но не обе одновременно. Еще одна особенность множественного выбора заключается в том, что вы обязательно должны выбрать одну из опций.

Группирование кнопок опций

Кнопки опций подразумевают исключительный выбор. Однако форма может содержать несколько групп кнопок опций. Сама система Windows не способна определить как связаны друг с другом различные кнопки опций. В Windows и в Delphi эта проблема решается так: связанные кнопки опций помещаются внутри компонента-контейнера. Для совместного хранения кнопок опций – и функционального и визуального – стандартный интерфейс пользователя Windows использует управляющий элемент блока группы. В Delphi этот управляющий элемент реализован в компоненте GroupBox. Аналогичный компонент, который используется для кнопок опций, - RadioGroup, представляющий собой блок группы с нарисованными внутри него несколькими копиями кнопок опций. Компонент RadioGroup способен автоматически выравнивать свои кнопки опций и вы легко можете добавить в него новые элементы на этапе выполнения.

Правила построения блока группы с кнопками опций очень просты: разместите в форме компонент GroupBox, а затем поместите в него кнопки опций. Компонент GroupBox способен содержать другие управляющие элементы и вместе с компонентом Panel является одним из самых распространенных компонентов-контейнеров. Если вы отключите или скроете блок группы, все управляющие элементы внутри него также будут заблокированы или скрыты.

Вы по-прежнему можете обрабатывать отдельные кнопки опций, но можете работать также со всем массивом управляющих элементов, принадлежащих блоку группы. Соответствующее свойство называется Controls. другое свойство ControlCount – хранит число элементов. К этим двум свойствам можно получить доступ только на этапе выполнения.

Задание № 5.2

Выполнить построитель английских предложений по типу The book is on the table. Цель данного прмера состоит в том, чтобы создать инструмент для построения подобных фраз путем выбора из различных доступных опций.

1) Поместите в форму компонент GroupBox с заголовком First Object и затем кнопки опций с заголовками The book, The pen, The pencil, The chair.
2) Поместите еще один компонент GroupBox с заголовком Position с опциями on, under, near.
3) Поместите компонент RadioGroup с заголовком Second Object с опциями the table, the big box, the carpet, the computer. В этом случае для создания элементов вы должны ввести список значений в свойство Items (тип TStringList).
4) Одну из кнопок опций в каждой группе нужно обязательно пометить, установив свойство Checked в True, соответствующее значение должно иметь и свойство ItemIndex группы опций (которое указывает на текущий выбор).
5) Поместите в верхнюю часть формы компонент Label, где будет отображаться построенная фраза.
6) Выберите в форме все кнопки опций (щелкая по каждой из них при нажатой клавише Shift) и введите имя метода TForm1.ChangeText в окне Object Inspector после события OnClick. Код этого метода представлен ниже.

```
procedure TForm1.ChangeText (Sender: TObject);
var
  Phrase:String;
  I:Integer;
begin
  For i:=0 to GroupBox1.ControlCount-1
   do if (GroupBox1.Controls[i] as TRadioButton).Checked
       then Phrase:= (GroupBox1.Controls[i] as TRadioButton).Caption;
  Phrase:= Phrase + ' is ';
  For i:=0 to GroupBox2.ControlCount-1
    do  with
          GroupBox2.Controls[i] as TRadioButton do
          if Checked
            then Phrase:= Phrase + Caption;
          Label1.Caption:= Phrase + ' ' + RadioGroup1.Items[RadioGroup1.ItemIndex];
end;
```

Список со многими опциями

Когда вам нужно добавить много опций, кнопки опций не подходят. Для решения этой проблемы используется список. Список способен хранить в небольшом месте большое количество опций и может содержать полоску прокрутки, чтобы показывать на экране только ограниченную часть всего выбора. Другое преимущество списка состоит в том, что вы легко можете добавить в него новые элементы или удалить некоторые из текущих. Списки чрезвычайно гибки и мощны.

Еще одна важная особенность: используя компонент ListBox, вы можете осуществлять как однозначный выбор – поведение, аналогичное группе кнопок опций, - так и множественный выбор – подобно группе чекбоксов.

Задание № 5.3

Выполнить построитель английских предложений, используя компонент ListBox.

1) Поместите в центре формы компонент RadioGroup с заголовком Position и опциями on, under, near (одну опцию пометьте).
2) Слева и справа от блока группы поместите два списка и добавьте несколько строк в свойства Items обоих списков. Вы можете скопировать все строки из редактора свойств Items одного списка и вставить в редактор такого же свойства другого списка. Для удобства строки можно отсортировать, установив свойство Sorted списков в True. Не забудьте также поместить над списками пару надписей, раскрывающих их содержание.
3) В верхнею часть формы поместите компонент Label, в котором будет отображаться построенная фраза.
4) В нижней части формы поместите поле редактирования с надписью и кнопку с заголовком Add . В поле редектирования будет вводиться строка текста, которую необходимо добавить в оба списка по нажатию кнопки Add.
5) Чтобы первоначально выбрать по строке из каждого списка, записать метод для события OnCreate формы:

```
procedure TForm1.FormCreate(Sender:TObject);
var
  N:Integer;
begin
   N:= ListBox1.Items.IndexOf('book');
   ListBox1.ItemIndex:= N;
```

```
   N:= ListBox2.Items.IndexOf('table');
   ListBox2.ItemIndex:= N;
end;
```

6) Создать процедуру ChangeText, соединенную с событиями OnClick группы опций и двух списков. Чтобы получить из списка выбранный текст, вы должны только взять номер выбранного элемента (хранится в свойстве этапа выполнения ItemIndex) и затем отыскать строку в соответствующей ячейке массива Item (ListBox1. Items [ListBox1. ItemIndex]).

7) Добавлять строку String1 в список с помощью метода Add класса TStrings – **`ListBox1.Items.Add(String1).`**

6. Создание и обработка меню.

Структура меню

Обычно меню имеет два уровня. Полоска меню, которая находится под заголовком окна, содержит имена выпадающих меню. Каждое выпадающее меню содержит несколько элементов. Однако структура меню очень гибка. Элемент меню можно поместить непосредственно в полоску, а выпадающее меню внутрь другого выпадающего меню. Выпадающее меню внутри другого выпадающего меню (выпадающее меню второго уровня) встречается довольно часто, и для этого случая Windows предоставляет специальный визуальный значок – маленький треугольник справа от меню. Нередко вместо выпадающего меню второго уровня вы можете просто сгруппировать ряд опций в первоначальном выпадающем меню и поместить два разделителя: один до группы и один после.

Различные роли элементов меню

Существует три основных типа элементов меню:

- Команды – элементы меню, которые используются для выдачи команды и выполнения действия. Визуально они никак не выделяются.
- Установщики состояния – элементы меню, которые используются для переключения опции в положения включено – выключено и изменения состояния какого – либо элемента. Если эти команды имеют два состояния, то в активном положении слева от них обычно стоит галочка. В этом случае выбор команды изменяет состояние на противоположное.
- Элементы вызова диалога – элементы меню, которые вызывают диалоговою панель. Реальное различие между этими и другими элементами меню состоит в следующем: с помощью этих элементов пользователь должен получить возможность исследовать вероятные действия соответствующей диалоговой панели. Такие команды должны иметь визуальный ключ в виде трех точек после текста.

Редактирование меню с помощью Menu Designer

Система Delphi включает специальный редактор для меню Menu Designer. Чтобы вызвать этот инструмент, поместите компонент меню в форму и щелкните по нему. Не волнуйтесь о точном положении данного компонента в форме, поскольку на результат это не влияет : само меню всегда помещается правильно – под заголовком формы. Menu Designer позволяет создавать меню путем простого написания текста команд, перемещать элементы или выпадающие меню с помощью буксировки и легко устанавливать свойства элементов. Для создания выпадающего меню второго уровня нужно выбрать команду Create submenu в SpeedMenu инструмента (локальном меню, которое вызывается правой кнопкой мыши).

Горячие клавиши меню

Общее свойство элементов меню – наличие подчеркнутой буквы. Эту букву можно использовать для выбора меню с помощью клавиатуры. При одновременном нажатии клавиши Alt и клавиши с буквой на экране появляется соответствующее выпадающее меню. Чтобы определить подчеркнутую клавишу, просто поместите перед ней символ амперсанта (&), например &File. Элементам меню можно назначить горячие клавиши. Для этого нужно указать значение для свойства ShortCut, выбрав одну из стандартных комбинаций.

Задание № 6.1

1) С помощью Menu Designer добавьте в форму полоску меню. Эта полоска имеет три выпадающих меню: меню File с единственной командой Exit; меню Options с командами Font, Color, Left, Center, Hight (между Color и Left установите разделитель, для Left, Center, Hight назначьте горячие клавиши); меню Help с элементом About. Чтобы поместить разделитель, вместо текста команды вставьте просто дефис.

2) В форму поместите компонент RichEdit и две пиктограммы диалогов FontDialog и ColorDialog.
3) Для отклика на команды меню вы должны определить метод для события OnClick каждого элемента меню. Код метода TForm1.Font1Click :

```
procedure TForm1.Font1Click(Sender:TObject);
begin
  FontDialog1.Font:= RichEdit1.Font;
  FontDialog1.Execute;
  RichEdit1.Font:= FontDialog1.Font;
end;
```

Код метода TForm1.Color1Click аналогичен предыдущему.
Код метода TForm1.Left1Click :

```
procedure TForm1.Left1Click(Sender:TObject);
begin
  RichEdit1.Paragraph.Alignment:= taLeftJustify;
  Left1.Checked:= True;
  Center1.Checked:= False;
  Right1Checked:= False;
end;
```

Чтобы поставить галочку в ряде выбираемых опций, установите свойство Checked элемента меню в окне Object Inspector в True.
4) Код методов для элементов Center и Right аналогичен предыдущему.

Изменение элементов меню

Для модификации элемента меню чаще всего используются три свойства. Свойство Checked используется, чтобы добавить или удалить галочку рядом с элементом меню. С помощью свойства Enabled элемент меню можно пригасить, после чего пользователь не сможет его выбрать. Последнее свойство этой группы Caption представляет текст элемента меню. Изменяя текст элемента меню, вы указываете пользователю, что программа перешла в другое состояние.

Задание № 6.2

1) На форме расположить две панели , две кнопки и компонент RichEdit. Первая панель содержит два поля редактирования, а вторая два чекбокса. Также необходимо построить выпадающее меню File, Buttons, View, Pulldowns. Команды меню File – Open , SaveAs. Команды меню Buttons содержит изменяемый текст (с ‘Enable First’ на ‘Disable First’). Команды меню View – Edit Boxes, Check Boxes. Команды Pulldowns – Remove File, Disable Buttons, Disable View. Поместите в форму пиктограммы необходимых диалогов.
2) Код методов, которые загружают и сохраняют файлы :

```
procedure TForm1.Open1Click(Sender:TObject);
begin
  if  OpenDialog1.Execute
    then RichEdit1.Lines.LoadFromFile OpenDialog1.FileName);
end;

procedure TForm1.SaveAs1Click(Sender:TObject);
begin
  if  SaveDialog1.Execute
    then RichEdit1.Lines.SaveToFile(SaveDialog1.FileName);
end;
```

3) Компоненты внутри панелей в действительности не используются. Однако вам необходимо воспользоваться двумя кнопками, чтобы скрыть или отобразить каждую из двух панелей вместе с управляющими элементами, которые в них содержатся. Те же действия можно выполнить с помощью двух команд меню : View/ Edit Boxes и View/ Check Boxes. Когда вы выбираете одну из этих команд меню или нажимаете одну из кнопок, происходит три разных действия. Во-первых, отображается или скрывается панель. Во-вторых, текст кнопки изменяется с Hide на Show, и наоборот. В-третьих, рядом с соответствующим элементом меню появляется или исчезает галочка. Ниже приведен код одного из двух методов, который связан с событиями щелчка как команды меню, так и кнопки :

```
procedure TForm1.ViewEdit1Click(Sender:TObject);
begin
   Panel1.Visible:= not Panel1.Visible;
   ViewEdit1.Checked:= not ViewEdit1.Checked;
   if Panel1.Visible
     then  Button1.Caption:='Hide';
       else Button1.Caption:= 'Show';
end;
```

4) Команды меню Buttons применяют другой подход. Для показа текущего состояния они используют не галочку, а изменение текста. Кроме того, они разрешают или запрещают соответствующую команду View и кнопку.

```
procedure TForm1.ButtonsFirst1Click(Sender:TObject);
begin
  if Buttons1.Enabled
    then begin
           Buttons1.Enabled:= False;
           ViewEdit1.Enabled:= False;
           ButtonsFirst1.Caption:= 'Enable &First';
        end
      else begin
            Buttons1.Enabled:= True;
            ViewEdit1.Enabled:= True;
            ButtonsFirst1.Caption:= 'Disable &First';
          End;
end;
```

5) Команды меню Pulldowns должны скрывать выпадающее меню указанные в элементах и показывать галочку для выбранного элемента. Запишите код для каждого элемента этого меню самостоятельно.

7. Получение ввода от мыши. Рисование в форме.

Когда пользователь нажимает одну из кнопок мыши, указатель которой находится над формой (или над компонентом), Windows посылает приложению несколько сообщений. Для написания кода, откликающегося на эти сообщения, Delphi определяет несколько событий. Основных событий два : OnMouseDown, которое происходит при нажатии одной из кнопок мыши, и OnMouseUp, которое происходит при освобождении кнопки.

Еще одно важное системное сообщение связано с перемещением мыши – сообщение OnMouseMove. Событие OnClick также доступно и в форме. Его основной смысл состоит в том, что левая кнопка мыши нажимается и отпускается над одним и тем же окном или компонентом. Однако в период между этими двумя действиями курсор может переместиться за пределы окна или компонента, причем левая кнопка мыши будет удерживаться нажатой. Если вы в определенной позиции нажмете кнопку мыши, а затем переместите мышь в другое место и отпустите кнопку, то щелчок не произойдет. В этом случае окно получает только сообщение о нажатии, несколько сообщений о перемещении и сообщение об освобождении.

События, связанные с мышью

Метод, соответствующий событию OnMouseDown имеет несколько параметров:

```
procedure TForm1.FormMouseDown(Sender:TObject;Button:TMouseButton;Shift:TShiftState;
  X,Y:Integer ) ;
```

Кроме обычного параметра Sender здесь присутствуют еще четыре :

1) Button – показывает, какая из трех кнопок мыши была нажата. Возможные значения : mbRight, mbLeft, mbCenter.
2) Shift – показывает, какие влияющие на мышь клавиши были нажаты при возникновении события. Такой клавишей может быть Alt, Ctrl или Shift, нажатая вместе с самой кнопкой мыши. Данный параметр имеет тип множества, т.к. несколько клавиш могут быть нажаты одновременно. Это означает, что при анализе условия вы должны применять не проверку на равенство, а оператор in.
3) X и Y – показывают координаты позиции мыши относительно клиентской области.

Рисование в форме

Canvas (холст) – это область рисунка в форме и многих других графических компонентах. Чтобы получить доступ к пикселам формы, используйте свойство формы Canvas и свойство Pixels для Canvas. Свойство Pixels – это двумерный массив, соответствующий цветам отдельных пикселов в Canvas. Canvas.Pixels[10,20] соответствует цвету пиксела, который находится на 10 пикселов правее и на 20 пикселов ниже точки отсчета. Обращайтесь с массивом пикселов как с любым другим свойством; чтобы изменить цвет пиксела присвойте новое значение. Чтобы определить цвет пиксела – прочитайте значение.

Каждое свойство Canvas имеет воображаемое перо для рисования линий и контуров. Свойство Pen (перо) определяет цвет и размер линий и границ фигур. Свойствами пера являются его цвет, размер (если это сплошная линия) или стиль. Работая с пером, вы можете прочитать (но не изменить) его текущую позицию (свойство PenPos). Позиция пера определяет исходную точку следующей линии, которую программа может нарисовать с помощью метода LineTo. Для изменения позиции вы можете применить метод MoveTo канвы.

Свойство Brush (кисть) определяет цвет очерченой поверхности. Кисть используется для закрашивания замкнутых фигур. Свойствами кисти являются ее цвет, стиль и иногда растровое изображение.

Свойство Font определяет шрифт, который используется методом холста TextOut для написания текста в форме. Шрифт имеет имя, размер, стиль, цвет и т.п.

Задание № 7.1

1) Поместить в форму меню Color с командами PenColor и BrushColor, которые будут соответственно изменять цвет пера и кисти с помощью стандартной диалоговой панели.
2) В форме реализовать рисование окружностей, эллипсов и прямоугольников различных размеров и цветов с помощью мыши, используя свойство Canvas формы. Дальше даются подсказки для реализации этой задачи.
3) Запишите следующий код для события OnMouseDown :

```
if Button = mbLeft
 then begin
        Center.X:= X;
        Center.Y:= Y;
        if ssShift in Shift
         then Circle:= False
           else  Circle := True;
      end;
```

Поле формы Circle типа Boolean определяет вид фигуры, значения координат центра фигуры записываются в поля формы Center типа TPoint.

4) Запишите следующий код для события OnMouseUp :

•

```
Radius.X:=abs(Center.X-X);
Radius.Y:=abs(center.Y-Y);
if Circle
  then Canvas.Ellipse(Center.X-Radius.X,Center.Y-Radius.Y,Center.X+Radius.X,Center.Y+
        Radius.Y)
   else Canvas.Rectangle(Center.X-Radius.X,Center.Y-Radius.Y,Center.X+Radius.X,
Center.Y+ Radius.Y);
```

5) Запишите следующий код для события OnMouseMove :

```
Caption:= Format('Координаты: x=%d, y=%d ',[X,Y]);
```

6) Запустите приложение. Если все сделано правильно, то вы будете наблюдать изменение координат в заголовке формы при продвижении мыши; сможете рисовать окружности и эллипсы нужного размера (щелкая кнопкой и удерживая ее при перемещении мыши по горизонтали и вертикали); сможете рисовать прямоугольник нужного размера, используя ту же технологию.

Черчение и рисование в системе Windows

- Черчение – вы обращаетесь к канве и вызываете некоторые ее методы. Поскольку изображение не сохраняется, форма может частично или целиком потерять свое содержимое (при закрытии окна формы другим окном или при уменьшении размера окна формы).
- Рисование – это технология, которая позволит приложению перерисовывать всю ее поверхность при любых возможных условиях.

  Для вызова перерисовки можно использовать методы Invalidate, UpDate, ReFresh и Repaint.

Задание № 7.2

1) Начиная с оператора if., код для события OnMouseUp перенести в код для события OnPaint.
2) В код для события OnMouseUp вставить в конце вызов метода Invalidate (который вызывает косвенно метод FormPaint, связанный с событием OnPaint).
3) Запустить приложение. При правильном выполнении всех инструкций, в форме будет рисоваться только одна фигура.
4) Выполните первое задание для компонента PainBox.
5) Изучите возможности компонента Shape.

8. Инструментальная линейка и строка состояния

Во многих приложениях Windows имеются инструментальная линейка в верхней части окна и строка состояния в его нижней части. Инструментальная линейка содержит обычно несколько маленьких кнопок, которые вы можете нажимать щелчком мыши, чтобы задавать команды или переключать опции вкл. и выкл. Иногда инструментальная линейка может содержать комбинированный список, строку редактирования или некоторые другие управляющие элементы.

Строка состояния обычно имеет одну или несколько областей с текстовым описанием текущего состояния программы. У вас может быть область для координат, для показа выбранного шрифта или отображения всплывающих подсказок о том, что делать дальше, сообщений об ошибках и т.д. Фактически приложение определяет, что должно войти в строку состояния.

Построение инструментальной линейки

Для создания инструментальной линейки или строки состояния в Delphi вы можете использовать компонент Panel, Добавив в него несколько кнопок или других панелей, или же можете использовать специальный компонент StatusBar.

Чтобы построить типичную инструментальную линейку, вам нужно поместить панель в верхней части формы и разместить в ней несколько компонентов SpeedButton (быстрая кнопка). Быстрые кнопки могут иметь заголовок и значок, хотя обычно они имет только графический элемент. Быстрые кнопки могут вести себя подобно командным кнопкам, чекбоксам или кнопкам опций (радиокнопкам) и иметь другие растровые изображения для различных ситуаций. Быстрые кнопки являются графическим элементом, они не имеют дескриптора окна (т.о. не использует ресурсы окон), не могут получать фокус, не участвуют в переборе клавишей Tab и быстрее создаются и закрашиваются.

Если вы просто выбираете компонент SpeedButton и помещаете его экземпляр в панель, то в результате получите графическую командную кнопку. Потом можно выбрать растровое изображение или ввести заголовок и записать код для события OnClick. Чтобы добавить группу быстрых кнопок, которые будут работать подобно радиокнопкам, поместите их в панель и присвойте их свойствам GroupIndex одинаковое значение. Все кнопки, имеющие одинаковое свойство GroupIndex, станут осуществлять взаимоисключающий выбор аналогично кнопкам опций. Одна из этих кнопок должна быть всегда выбранной, поэтому не забывайте присвоить свойству Down значение True для одной из них на этапе проектирования или как только программа начинает работать.

В качестве альтернативы у вас могут быть взаимоисключающие кнопки, которые все могут быть ненажатыми. Это означает, что вы можете щелкнуть на выбранной кнопке и отменить ее выбор. Вы можете выбрать этот режим, присвоив свойству AllowAllUp для всех кнопок группы значение True. Чтобы быстрая кнопка работала как чекбокс необходимо выполнить следующее. Чекбокс – это группа только с одним эелементом, в котором все кнопки могут быть невыбранными. На практике вы добиваетесь этого добавив новую быструю кнопку, присвоив ей конкретное значение для свойства GroupIndex (отличное от индексов других групп, и выбрав значение True для свойства AllowAllUp..

Задание № 8.1

Постройте инструментальную линейку, в которой будут располагаться следующие кнопки:

1) командная, при нажатии которой будет издаваться звук;
2) группа из трех кнопок, которые будут осуществлять выравнивание текста в компоненте Label по левому, правому краю и по центру и вести себя как кнопки опций;
3) группа из трех кнопок, которые будут изменять стиль текста компонента Label (полужирный, курсив, подчеркнутый) и могут быть все ненажатыми – в этом случае текст нормальный (вы должны не просто выбрать стиль, когда нажата кнопка , но и отменить ее выбор, восстанавливая нормальный стиль, когда кнопка освобождена. В методах, отвечающих за событие щелчка этих кнопок, вы должны написать:

```
procedure TForm1.SpeedButton1Click(Sender:TObject);
begin
  if SpeedButton1.Down
    then Label1.Font.Style:= [fsBold]
     else Label1.Font.Style:= []
end;
```

4) кнопка, которая будет изменять размер шрифта с 24 на 12 и вести себя как чекбокс, т.е. при нажатой кнопке размер шрифта 24, при отжатой – 12.
5) Добавьте в форму меню, с помощью которого можно будет отключать и включать различные кнопки используя свойство Enabled , скрывать и показывать линейку с помощью свойства Visible панели.

Добавление всплывающих подсказок в инструментальную линейку

Всплывающая подсказка – текст, который кратко описывает быструю кнопку под курсором. Этот текст обычно выводится на экран в окне желтого цвета, после того как курсор мыши оставался неподвижным над кнопкой в течение некоторого времени. Для добавления всплывающих посказок в инструментальную линейку приложения просто устанавливают значение свойства ShowHints панели, которая используется как инструментальная линейка, равным True. Вы также можете изменить значение по умолчанию свойств HintColor и HintPause глобального объекта Application и, что необязательно, добавить непосредственно конкретный текст для всплывающей подсказки кнопок инструментальной линейки (свойство Hint). Текст подсказок можно изменять при изменении состояния кнопки написав некоторый код, использующий свойство Hint.

Комбинированный список в инструментальной линейке

Существует несколько распространенных приложений, которые используют комбинированные списки в инструментальных линейакх для того, чтобы показывать списки стилей, шрифтов, размеров шрифта и т.д. Используем в следующем задании такой подход: создадим комбинированный список, чтобы позволить пользователю выбирать шрифт и окно редактирования с прокруткой компонент (SpinEdit) для установки размера шрифта, каждый с собственной меткой. Два новых компонента и их метки имеют соответствующее сообщение в свойстве Hint. Комбинированный список заполняется, когда начинает работать приложение, копируя названия текущих шрифтов из объекта Screen.

```
procedure TForm1.Form1Create (Sender:TObject);
var
   I:Integer;
begin
  for I:= 1 to Screen.Fonts.Count do
    {скопируйте название шрифта в комбинированный список}
    ComboBox1.Items.Add Screen.Fonts.Strings[I-1]);
    {выберите текущий шрифт}
    ComboBox1.ItemIndex:=ComboBox1.Items.IndexOf(Label1.Font.Name);
end;
```

Когда выбирается новый элемент комбинированного списка – текст текущего элемента комбинированного списка копируется в название шрифта надписи.

```
procedure TForm1.ComboBox1Change(Sender:TObject);
begin
  Label1.Font.Name:= ComboBox1.Items[ComboBox1.ItemIndex];
end;
```

Для управляющего элемента редактирования с прокруткой :

```
procedure TForm1.SpinEdit1Change Sender:TObject);
begin
  if not (SpinEdit1.Text = ' ')
    then Label1.Font.Size:= SpinEdit1.Value;
end;
```

Задание № 8.2

1) Из формы, созданной в предыдущем примере, удалить кнопку изменения размера шрифта
2) Согласно вышеизложенному поместить в форму комбинированный список, чтобы позволить пользователю выбирать шрифт и окно редактирования с прокруткой компонент (SpinEdit) для установки размера шрифта, каждый с собственной меткой.
3) Записать код для добавленных компонентов.

Построение строки состояния

Система Delphi Содержит специальный компонент StatusBar, основанный на специальном управляющем элементе системы Windows 95. Этот коипонент может быть использован почти как панель, когда значением его свойства SimplePanel является True. В этом случае вы можете использовать свойство SimpleText, чтобы вывести некоторый текст. Этот компонент позволяет вам определить несколько подпанелей с помощью редактора свойства Panels. Каждая подпанель имеет свои собственные графические атрибуты, настраиваемые с помощью редактора. Другой характеристикой компонента строки состояния является специальная область, которая добавляется в нижний правый угол линейки и полезна для изменения размера самой формы.

Строка состояния используется для многих целей. Наиболее часто ее используют для вывода на экран информации об элементе меню, который в настоящее время выбран пользователем. При этом нужно сделать два шага. Сначала ввести строку как свойство Hint каждого элемента меню. Затем записать некоторый код для обработки события OnHint приложения. Вам нужно добавить вручную новый метод в форму и затем его присвоить свойству OnHint объекту Application при запуске :

```
procedure TForm1.Form1Create(Sender:TObject);
begin
  Application.OnHint:= ShowHint;
End;
```

В интерфейсной части кода формы вы можете добавить следующее определение

```
procedure TForm1.ShowHint(Sender: Object);
```

Эта процедура копирует текущее значение свойства Hint приложения, которое временно содержит копию всплывающей подсказки выбранного элемента, в строку состояния :

```
procedure TForm1.ShowHint(Sender:TObject);
begin
  StatusBar1.Panels[0].Text:=Application.Hint;
end;
```

Задание № 8.3

Сконструировать строку состояния, в которую вывести подсказки для всех команд всплывающего меню формы, построенной в предыдущих заданиях.

9. Приложение с несколькими формами. Многостраничные формы.

Обычно приложения имеют основное окно, несколько плавающих инструментальных панелей (или палитр) и диалоговых панелей, которые могут вызываться с помощью команд меню или командных кнопок.

Диалоговые панели в Delphi основаны на формах. С диалоговой панелью пользователь обычно связывает понятие модального окна. Модальное окно – это такое окно, которое получает фокус и должно быть закрыто прежде, чем пользователь может перейти обратно к основному окну. Это верно для панелей сообщения и диалоговых панелей. В системе Delphi можно также иметь немодальные диалоговые панели и модальные формы. Мы должны учитывать два момента:

1) рамки формы и пользовательский интерфейс определяют, выглядит ли эта форма как диалоговая панель
2) использование двух функций – Show и ShowModal для вывода на экран второй формы определяет поведение последней (немодальная или модальная).

Если вы используете функцию Show, вторая форма будет выведена на экран как немодальная. Код просто выводит форму на экран, а не создает ее. Форма создается файлом проекта. Первая созданная форма становится основной формой приложения. Когда модальная форма создается и выполняется с помощью функции ShowModal, она остается активной, до тех пор пока вы не закроете ее. Функция ShowModal не возвращает значение до тех пор, пока форма не закрыта. В это время основная форма остается недоступной. Как только модальная форма будет закрыта, функция ShowModal завершает работу, а код удаляет объект из памяти.

Добавление второй формы в программу

Для добавления второй формы можно нажать кнопку New Form или использовать команду File/New, переместиться на страницу Forms или Dialogs и выбрать один из доступных шаблонов формы. Если у вас в проекте две формы, используйте кнопку Select Form или Select Unit инструментальной линейки, чтобы управлять формами. Вы также можете выбрать, какая форма является основной и какие формы нужно создать автоматически при запуске,

используя страницу Forms диалоговой панели Project Options. Чтобы компилировать код первой формы, надо включить модуль, содержащий вторую форму (Unit2), с помощью оператора uses в список модулей в начале кода. Или можно выбрать первую форму и выполнить команду File/Use Unit. Чтобы закрыть вторую форму, вы можете использовать ее системное меню или нажать кнопку Close, помещенную в форму. При этом Delphi не закрывает вторую форму, а лишь скрывает ее, поэтому всегда есть возможность вывести на экран вторую форму.

Создание диалоговой панели

Чтобы построить диалоговую панель вместо формы, выберите значение bsDialog для свойства BorderStyle формы. После этого форму можно вывести на экран как модальное или немодальное окно, используя Show или ShowModal. Модальные диалоговые панели распространены больше, чем немодальные. Модальных форм следует избегать, т.к. пользователь не захочет иметь с ними дело. Вот один из подходов использования диалоговой панели :

1) Устанавливать начальные значения, каждый раз когда вы исполняете диалоговую панель.
2) Показывать диалоговую панель.
3) Если была нажата кнопка OK, скопировать новые значения обратно в форму.

Задание № 9.1

1) Подготовить две формы, в каждую из которых поместить по кнопке. В первой форме кнопка используется, чтобы показать второе окно, во второй форме – чтобы себя закрыть. Чтобы выполнить вторую форму запишите код:

```
procedure TForm1.Button1Click(Sender:TObject);
begin
  Form2.Show;
end;
```

2) Создадим форму, которая имеет две кнопки, используемые для создания модальной и немодальной форм. Как только вы создаете две новые формы ModalForm и ModalessForm, можно написать следующий метод, соответствующий событию OnClick одной из двух кнопок:

```
procedure TForm1.ModalButtonClick(Sender:TObject);
var
  Modal:TModalForm;
begin
   Modal:=TModalForm.Create Application);
   Modal.ShowModal;
   Modal.Free;
end;
```

Код для другой кнопки аналогичен, с той лишь разницей, что используется функция Show. Запишите его самостоятельно.

3) В форму поместить две надписи и кнопку. При нажатии кнопки должна появляться диалоговая панель с двумя растровыми кнопками OK , Cancel и два чекбокса, которые позволяют показывать и скрывать надписи формы. Записать следующий код :

```
procedure TForm1.Button1Click(Sender:TObject ;
var
  old1,old2:Boolean;
begin
  old1:= Form2.CheckBox1.Checked;
  old2:= Form2.CheckBox2.Checked;
  if (Form2.ShowModal = mrOk)
    then begin
           Label1.Visible:= Form2.CheckBox1.Checked;
           Label2.Visible:= Form2.CheckBox2.Checked;
         end
      else begin
             Form2.CheckBox1.Checked:= old1;
             Form2.CheckBox2.Checked:= old2;
           end;
end;
```

Построение блокнотов

Компоненты PageControl, TabControl (TabSet) и TabSheet используются для обработки страниц и ярлычков. TabSheet используется внутри компонента PageControl. Компонент TabControl (TabSet) используется для построения автономных ярлычков (не связанных со страницами). С помощью компонента PageControl можно построить приложение или диалоговую панель, используя понятие многостраничности или блокнота. Для добавления новых страниц (табличных листов) используется локальное меню компонента PageControl. Каждый объект TabSheet имеет собственный заголовок, который выводится на экран как ярлычок. Переключение между страницами на этапе

проектирования можно производить с помощью локального меню компонента PageControl или щелчком на ярлычке. Если вы поместите компонет на странице блокнота, то он будет доступен только на этой странице. Чтобы использовать один компонент для всех страниц, его необходимо поместить в форму вне компонента PageControl (или до выравнивания его с областью клиента) и затем переместить его на передний план, вызвав команду Bring to Front из локального меню формы. Допустим мы поместили в форму две кнопки, которые позволяют передвигаться по страницам : Next, Previous – которые представляют альтернативу использования ярлычков. Код, связанный с одной из кнопок :

```
procedure TForn1.BitBtnClick(Sender:TObject);
begin
  PageControl1.SelectNextPage(true);
end;
```

Другая кнопка вызывает ту же процедуру, передавая значение false в качестве параметра для выбора предыдущей страницы. Свойство SelectNextPage рассматривает последнюю страницу как единственную перед первой и перемещение происходит непосредственно между этими страницами. Существует третий метод, доступный для изменения страниц (после ярлычков и кнопок). Компонент список (ListBox) заполняется с помощью метода FormCreate путем копирования заголовка каждой страницы (свойство Page хранит список объектов TabSheet).

```
for I:= 0 to PageControl1.PageCount-1
   do ListBox1.Items. Add( PageControl1.Page[I].Caption;
```

Блокнот с набором ярлычков

Стандартный подход к построению блокнота с ярлычками в Delphi 1 использует два раздельных компонента NoteBook и TabSet. На этапе проектирования можно работать на страницах блокнота, изменяя значение его свойства PageIndex ; вы не можете щелкать на ярлысках, как с компонентами PageControl. Как только вы вводите новое значение для свойства PageIndex или ActivePage в Object Inspector, соответственно изменяется видимая страница блокнота. Можно также выбрать локальное меню блокнота, в котором имеются команды для перемещения к следующей и предыдущей странице.

Альтернативой использования PageControl, TabSet и NoteBook является использование компонента TabbedNoteBook.

Задание № 9.2

1) Поместите в форму компонент NoteBook, а затем компонент TabSet и выровняйте последний с нижней частью формы (alButtom), а компонент NoteBook с клиентской областью (alClient). Присвойте названия страницам блокнота, выбирая свойство Pages и вводя некоторые значения в соответствующем редакторе. Подготовить ярлычки с помощью ввода некоторых строк для свойства Tabs.
2) Чтобы соединить блокнот с множеством ярлычков нужно записать код для события OnChange компонента TabSet.

```
procedure TForm1.TabSet1Change(Sender:TObject;NewTab:Integer; var
AllowChange:Boolean);
begin
  NoteBook1.PageIndex:= NewTab;
end;
```

Вы можете активизировать страницу, используя также их названия (свойство ActivePage блокнота) вместо индекса (свойство PageIndex). Это работает, если только названия страниц совпадают с названиями ярлычков.

```
NoteBook1.ActivePage:=TabSet1.Tabs[NewTab];
```

Задание № 9.3

1) Поместите компонент TabbedNoteBook в форму. Создайте три страницы. Поместите кнопку Close, которая должна присутствовать на всех страницах.
2) На первой странице блокнота поместите список (ListBox) с названиями страниц. Чтобы изменить страницы в блокноте с помощью ListBox написать следующий код :

```
procedure TForm1.ListBox1Click(Sender:TObject);
begin
  TabbedNoteBook1.PageIndex:=ListBox1.ItemIndex;
end;
```

3) На второй странице блокнота поместите компонент FontDialog и кнопку для изменения шрифта. Чтобы изменить шрифт, когда пользователь щелкнет по соответствующей кнопке запишите код:

```
if FontDialog1.Execute
   then TabbedNoteBook1.TabFont:= FontDialog1.Font;
```

Задание № 9.4

1) В форму поместим компонент DriveComboBox и DirectoryListBox. На правой стороне формы поместить компонет Image и под ним TabSet.
2) Когда пользователь выбирает новый каталог, ярлычки должны сразу вывести на экран имена файлов с растровыми изображениями в ярлычки каждый раз, когда изменяется выбор в списке каталога. Для этого

поместите компонент FileListBox в форму и поместите его позади других компонентов или присвойте его свойству Visible значение False. Затем измените свойство Mask списка файлов на *.bmp. Для того чтобы связать компонент DirectoryListBox с компонентом FileListBox запишите в окне Object Inspectorдля свойства DirectoryListBox следующее имя компонента: FileListBox1. Теперь добавьте метод для обработки события OnChange списка каталога

```
procedure TForm1.DirectoryListBox1Change(Sender:TObject);
begin
  with FileListBox1
    do if Item.Count = 0
         then begin
                TabSet1.Tabs.Clear;
                Image1.Visible:= False;
                TabSet1.Tabs.Add('None');
              end
          else begin
                 Image1.Visible:= True;
                 TabSet1.Tabs:= FileListBox1.Items;
                end;
end;
```

3) Когда ярлычки выводят на экран имена файлов, для загрузки соответствующего растрового изображения в компонент Image используется следующий код :

```
procedure TForm1.TabSet1Change(Sender:TObject;NewTab:Integer;var
AllowChange:Boolean);
begin
   if TabSet1.Tabs[NewTab] <> 'None' then TabSet1.Tabs[NewTab];
end;
```

.

Контрольные вопросы ПЗ1 (ПК-1):

1. Как указать в программе ***Проводник*** директорию, содержащую все файлы проекта?(ПК-1).
2. Как добавить в проект еще одну форму и указать связанный с ней файл-модуль? (ПК-1).
3. Как удалить из проекта форму (с помощью ***Project Manager***)?(ПК-1).
4. Назовите наиболее распространенные операционные системы, в том числе – с открытым исходным кодом.(ПК-1).
5. Дайте определение операционной системы.(ПК-1).
6. Каковы цели работы операционной системы?(ПК-1).
7. Назовите компоненты компьютерной системы (включая программное обеспечение и пользователей).(ПК-1).
8. Назовите основные виды компьютерных систем, различающиеся по своему назначению и параметрам.(ПК-1).
9. Назовите основные архитектуры компьютерных систем и кратко определите, в чем суть каждой из них.(ПК-1).
10. Каковы основные компоненты операционной системы?(, ПК-1).
11. Каким образом происходило обращение к памяти и к внешним устройствам для ранних моделей компьютеров, при отсутствии операционных систем?(ПК-1).
12. Назовите классические ОС 1960-х – 1970-х гг., зарубежные и отечественные.
13. Каковы основная цель и идея разработки ОС UNIX?(ПК-1).
14. Назовите ОС для 8-разрядных, 16-разрядных и современных ПК.(ПК-1).
15. Какая, по Вашему, ОС является наиболее распространенной в мире?(ПК-1).
16. Назовите известные Вам диалекты ОС UNIX.(ПК-1).
17. Каковы основные возможности отечественной ОС ДИСПАК и для каких компьютеров она была разработана?(ПК-1).
18. Какие оригинальные идеи были положены в основу системы "Эльбрус" и ее операционной системы?(ПК-1).

ПРАКТИЧЕСКАЯ РАБОТА № 2.

Изучение системы Windows Server 2008

Обзор Windows Server 2008

Windows Server 2008- новая серверная *операционная система* фирмы Microsoft. Ее характерными особенностями являются расширенные возможности поддержки *параллельных вычислений* и управления сервером. Пользовательский *интерфейс Windows* 2008 во многом аналогичен интерфейсу клиентской ОС *Windows* 7 (см. лабораторную работу номер 4), т.е. более удобен, чем *интерфейс* предыдущих версий *Windows*. Система также требует значительно меньше времени и других ресурсов для запуска и эксплуатации, по сравнению с более ранними версиями *Windows*.

Существенно новая возможность *Windows* 2008 – новый мощный *командный процессор Windows* PowerShell.

Запуск системы

Включите *компьютер* с инсталлированной *Windows* 2008.

Через 0.5 – 1 мин. (примерное время загрузки системы) на экране появится *баннер* "Microsoft *Windows Server* 2008", затем – страница с приглашением нажать Ctrl – *Alt* – *Del* для входа в систему, после нажатия этой комбинации клавиш - стартовая страница для входа с именами пользователей (рис. 35.1).

Вход в систему и аутентификация пользователя

Выберите Ваше *имя пользователя* и кликните мышкой по картинке рядом с именем. Как правило, в систему уже введено стандартное имя *User*. Если для пользователя установлен *пароль*, введите его.

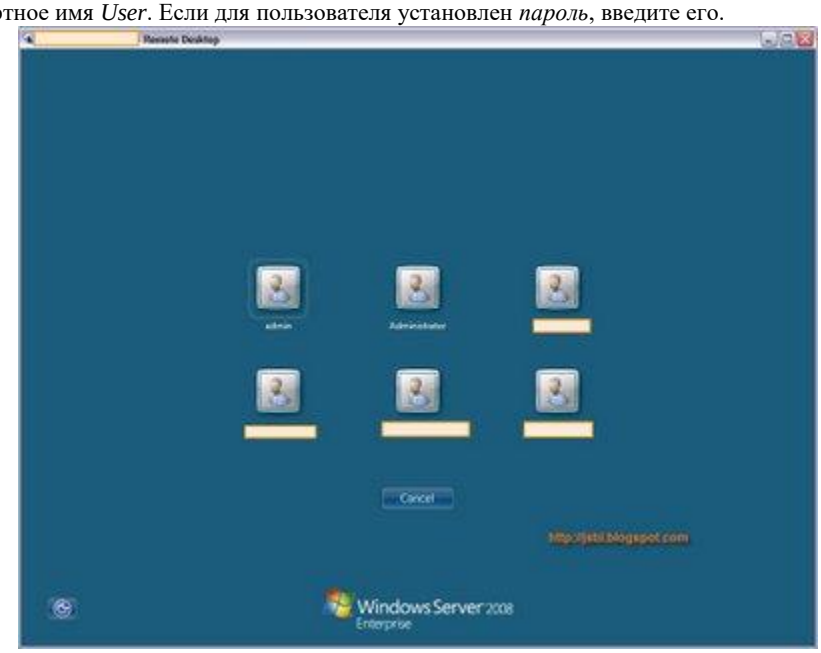
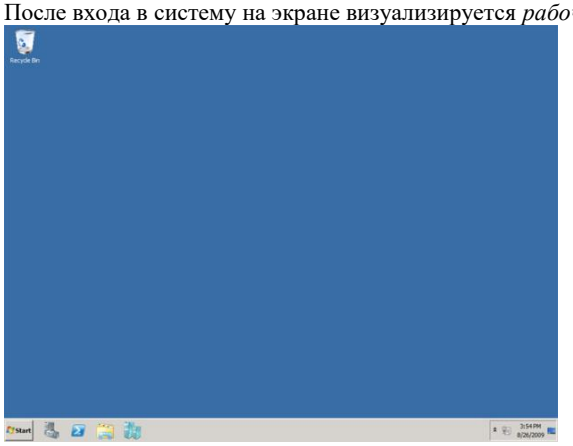


Рис. 35.1. Стартовое меню для входа пользователей в систему.

После входа в систему на экране визуализируется *рабочий стол* (desktop) (рис. 35.2) :



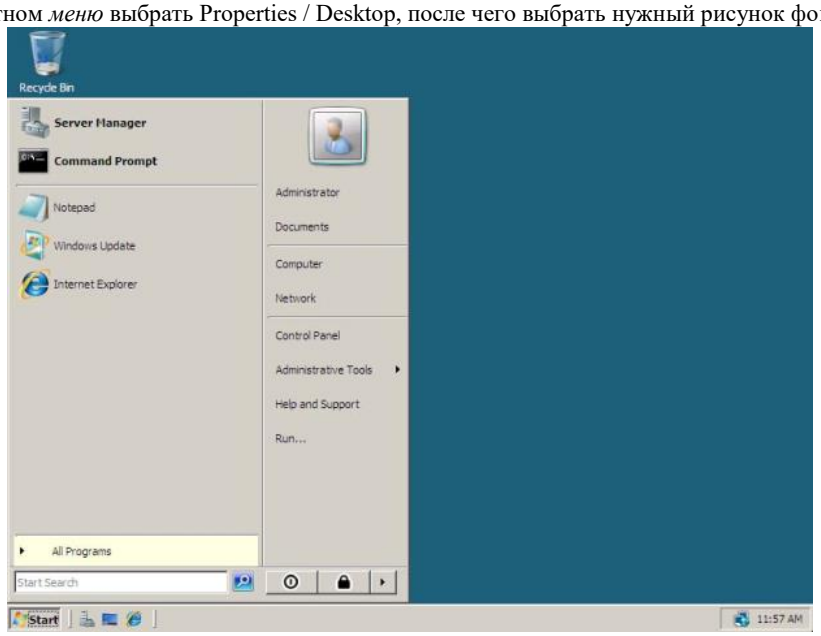
увеличить изображение

Рис. 35.2. Рабочий стол Windows 2008

Структура рабочего стола, мой компьютер, панель управления

Рабочий стол состоит из иконок приложений (например, *Internet* Explorer) и панели задач (taskbar) – обычно синего цвета, в нижней части. В левом нижнем углу расположена кнопка Start, при нажатии на которую *пользователь* может выбрать начальное действие – *запуск* какого-либо приложения, создание документа и др. (рис. 35.3).

Вид и фон рабочего стола при разных настройках могут отличаться. На рис. 35.2 показан один из возможных фонов рабочего стола. Для изменения фона рабочего стола необходимо на фоновом рисунке нажать правую кнопку мыши и в контекстном *меню* выбрать Properties / Desktop, после чего выбрать нужный рисунок фона в выпадающем списке.

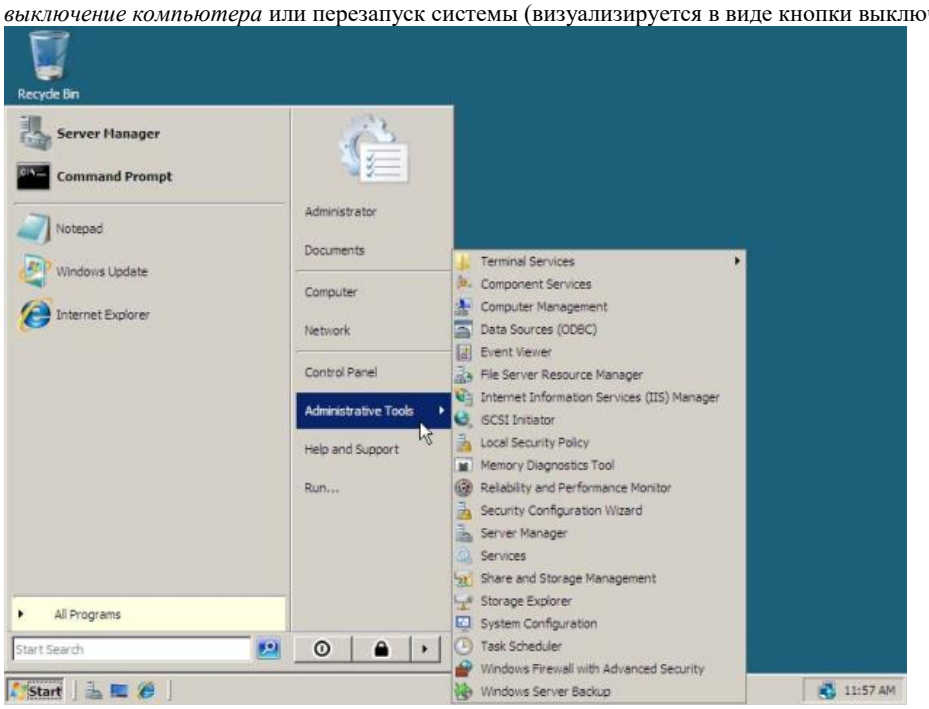


увеличить изображение

Рис. 35.3. Состояние рабочего стола после нажатия кнопки Start

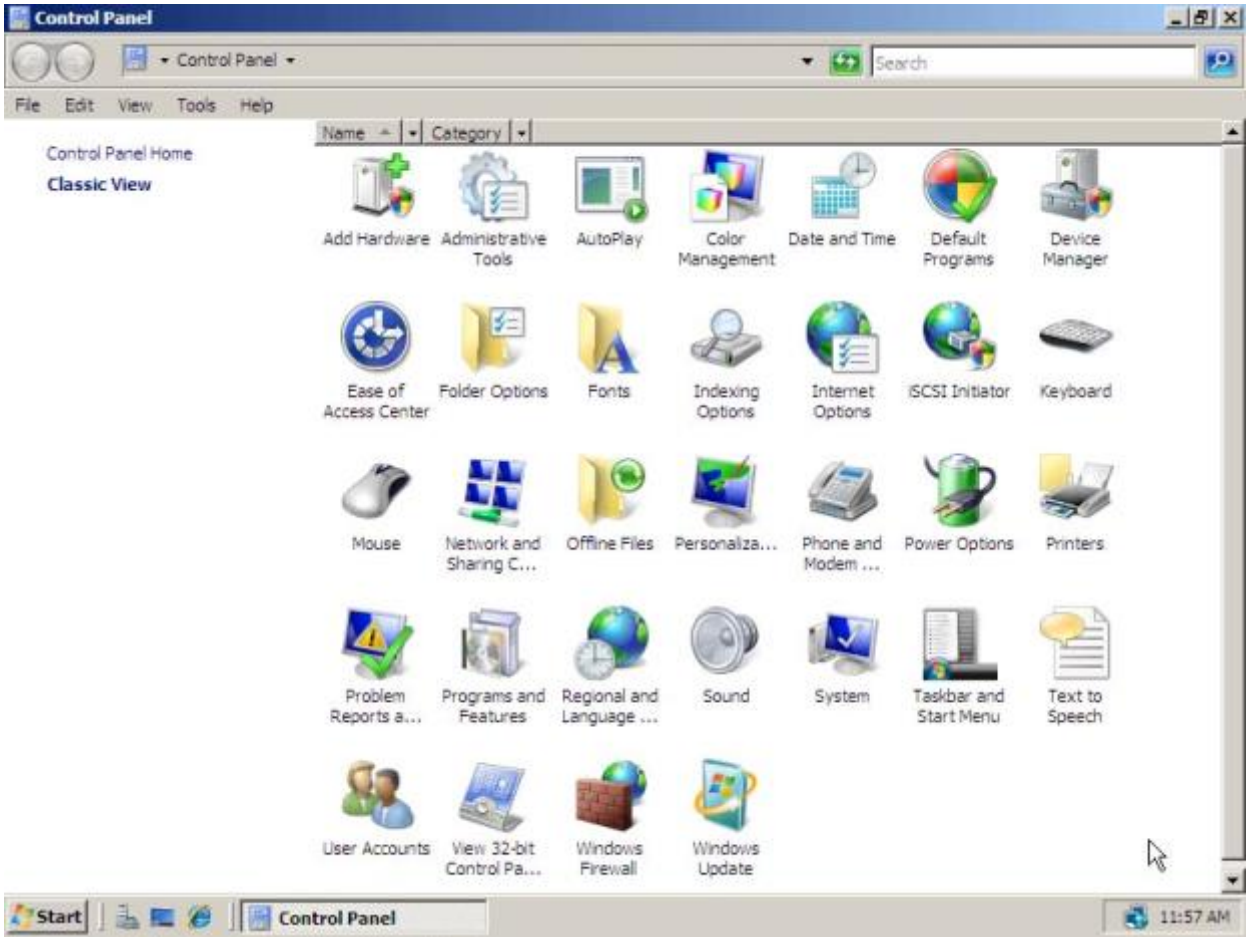
Основные пункты стартового *меню*, визуализируемого в результате нажатия кнопки Start:

- Computer – информация о компьютере, его ресурсах, устройствах, имени, установленной на нем ОС
- Documents – стандартная папка для создаваемых документов (Вы можете помещать документы и в любую другую более удобную Вам папку)
- Network– узлы локальной сети, доступные с компьютера
- *Administrative Tools* – инструменты для администрирования системы (рис. 35.4)
- *Control Panel* – панель управления (рис. 35.5)
- Run – запуск программ по именам
- Start Search – окно для поиска файлов
- (в нижней части) *Shut* Down / Log Off – выход из Вашего пользовательского сеанса, *выключение компьютера* или перезапуск системы (визуализируется в виде кнопки выключения питания).



увеличить изображение

Рис. 35.4. Administrative tools – утилиты для администрирования Windows Server 2008



увеличить изображение

Рис. 35.5. Панель управления

Рассмотрите более подробно панель управления (рис. 35.5). Она позволяет управлять ресурсами компьютера. Например, *пункт* Programs *and Features* позволяет инсталлировать новые программы, деинсталлировать или инсталлировать вновь ("ремонтировать") уже установленные.

Информация о компьютере и его папках и диска визуализируется при выборе Start / *Computer* рис. 35.6) программой *Windows* Explorer:

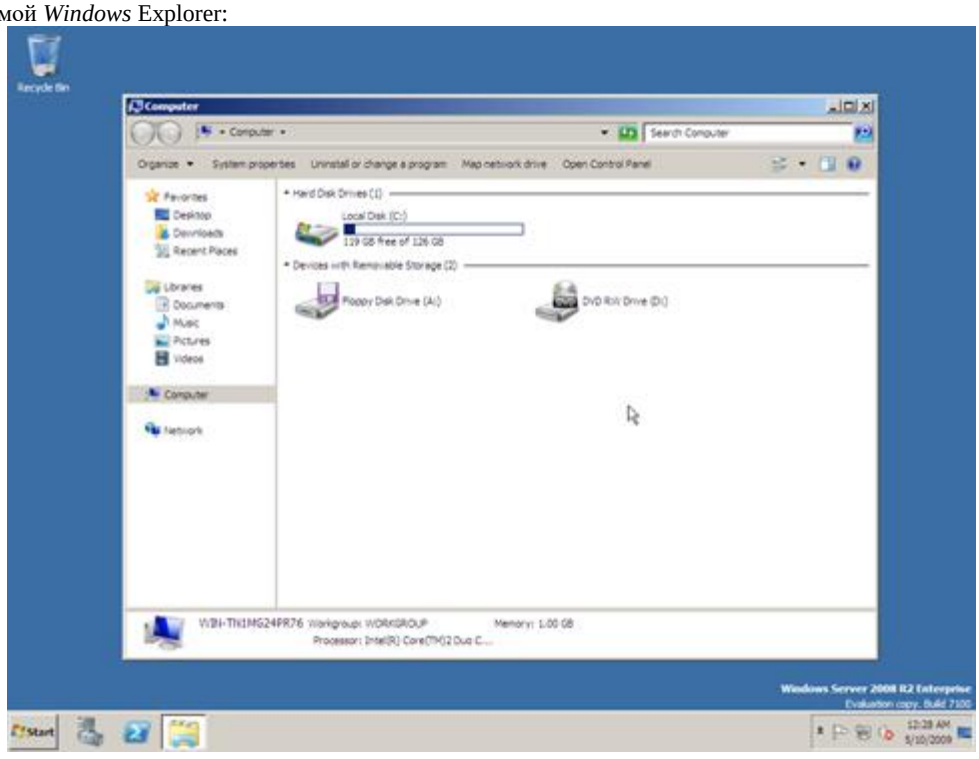
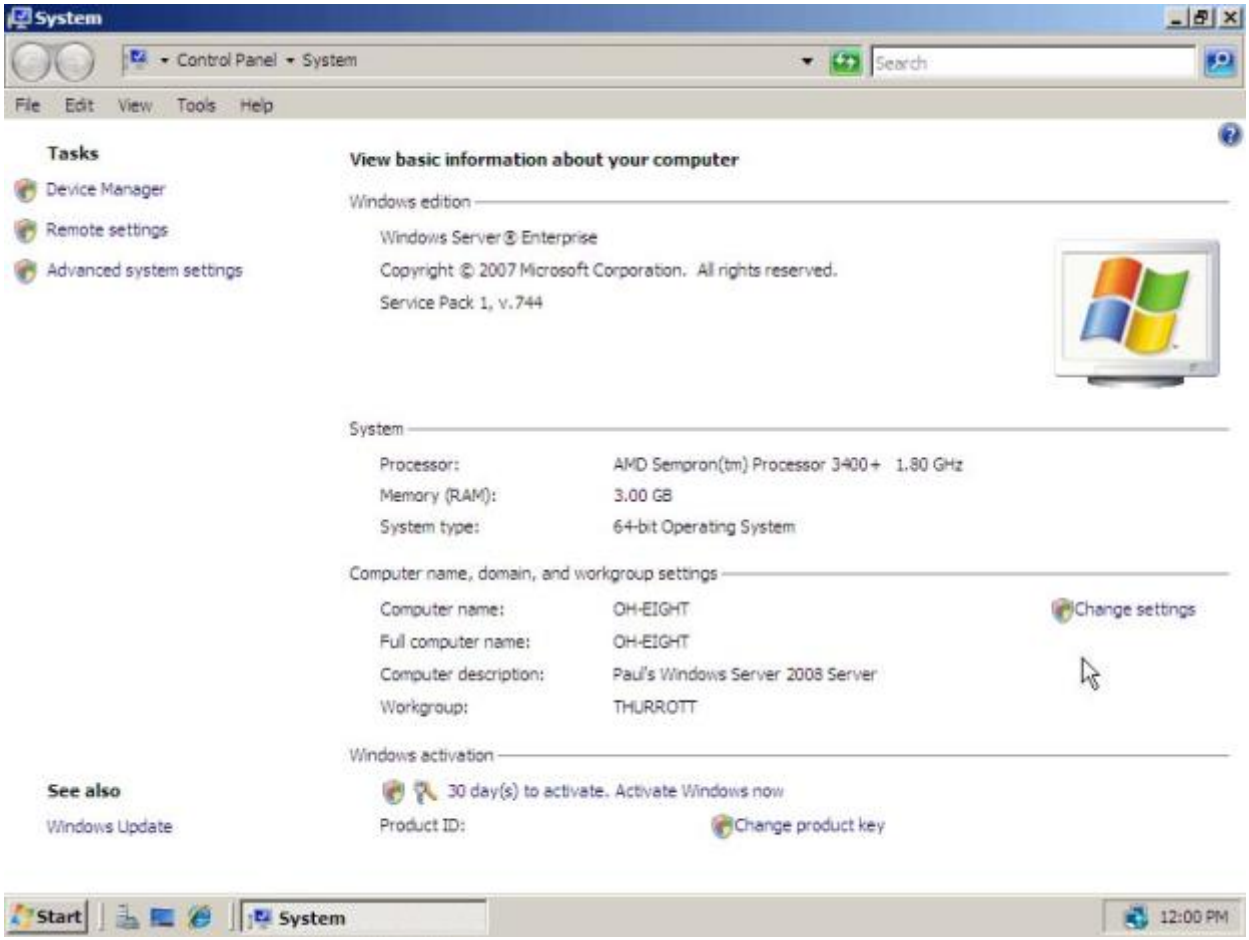


Рис. 35.6. Информация о папках и дисках компьютера (Windows Explorer)

Для визуализации основных свойств компьютера (системной информации) выберите в стартовом *меню*: Start / *Control Panel* / *System*. Возникает окно с системной информацией (рис. 35.7):



увеличить изображение

Рис. 35.7. Системная информация о компьютере

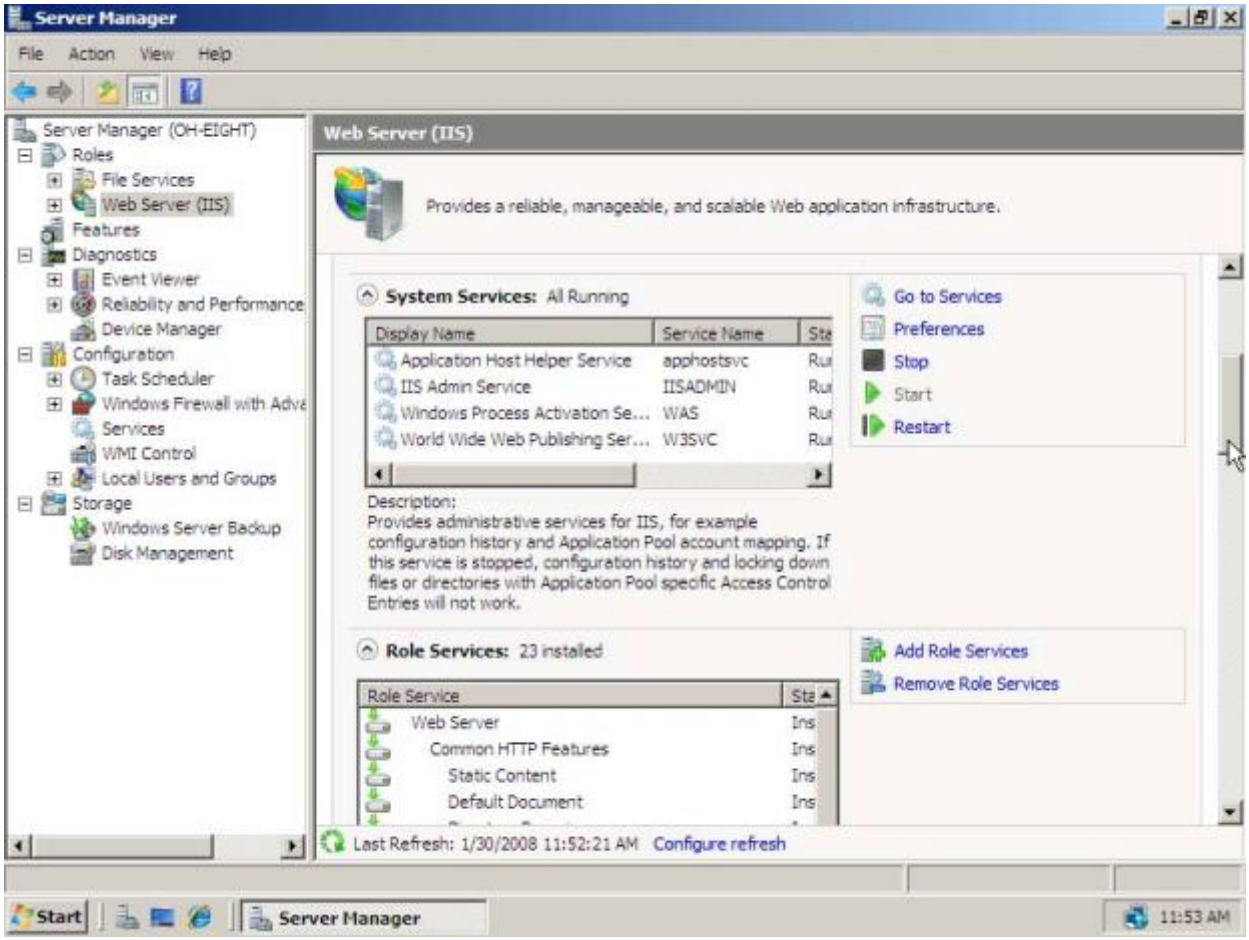
Вы видите информацию об ОС, объеме памяти, типе процессора и ряд вкладок, например, *Computer Name*, кликнув на которой, получаете информацию об имени компьютера. Кликнув *Device Manager*, получите подробную информацию о составе оборудования компьютера и установленных драйверах.

Работа с файлами и папками. Управление сервером

Работа с файлами и папками (folders) – хранилищами ссылок на файлы и другие папки – осуществляется с помощью программы *Windows* Explorer. На рис. 35.6 показано окно программы *Windows* Explorer, визуалилизующее информацию о дисках и основных папках компьютера. Если дважды подряд кликнуть на диске C:, то визуализируется содержимое его корневой папки, которая может содержать другие папки, и т.д. С помощью стрелки можно вернуться вверх на родительскую директорию.

Выбор файла или папки в директории осуществляется одним *кликом* мышки, вход в директорию или *открытие файла* – двойным *кликом* мышки на имени директории или файла. При этом для файла выполняется действие его открытия, зависящее от его типа, - для текстовых файлов – вызов соответствующего редактора (notepad, WordPad, MS *Word* и др.), для файлов *.pdf* – вызов Adobe Acrobat, для *исполняемых кодов* или командных файлов – *запуск* соответствующей программы или скрипта и т.д. Поэкспериментируйте на своем компьютере с навигацией по файлам и папкам и открытием файлов с документами.

Для управления сервером используется компонента *Server Manager* (Start / *Server Manager*) – см. рис. 35.8 .



увеличить изображение

Рис. 35.8. Server Manager – управление сервером в Windows 2008

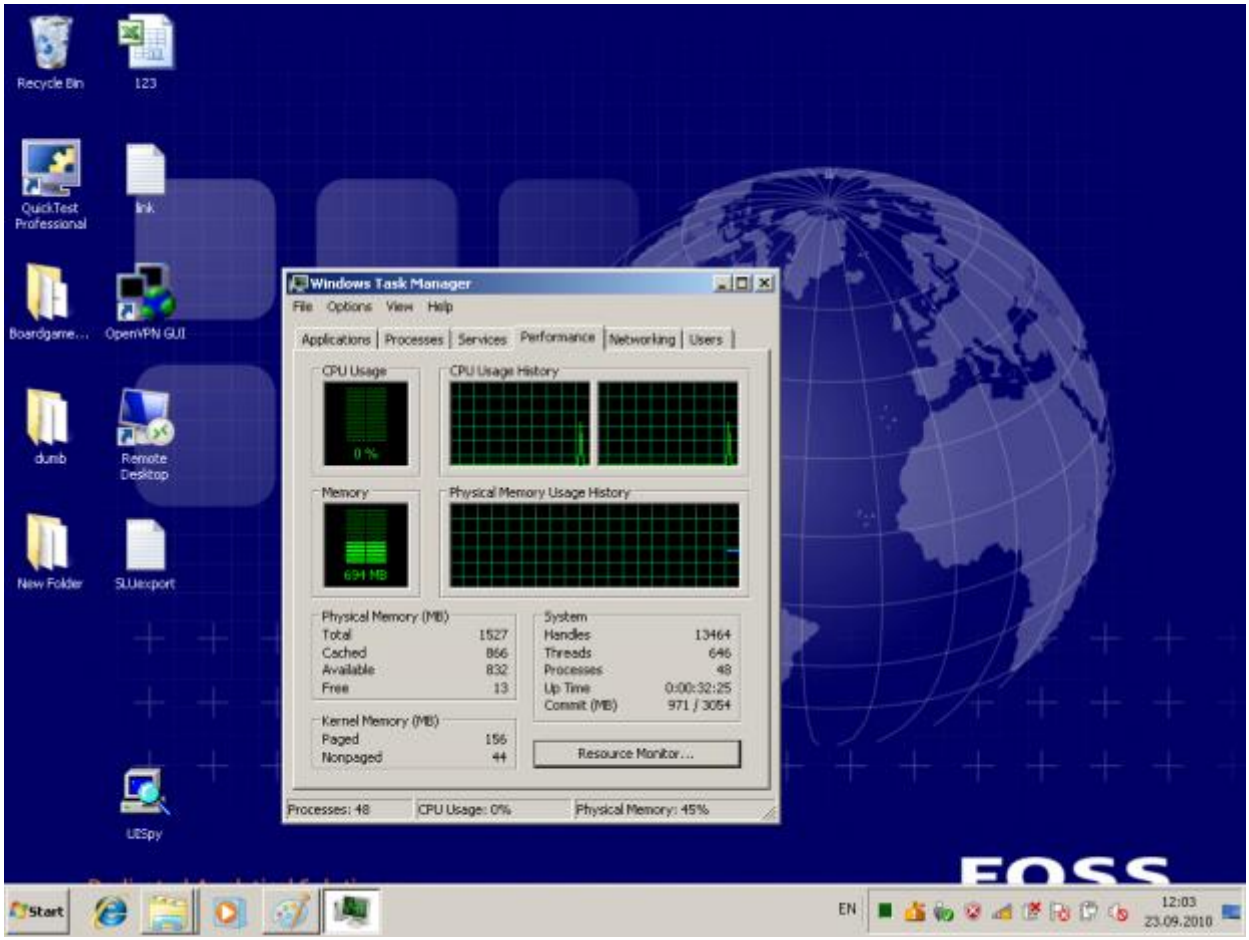
Запуск программ, управление задачами, программами и процессами

Есть несколько способов запустить программу:

- из Windows Explorer – дважды кликнуть на имени ее файла;
- из пункта меню Start – выбрать пункт Run, затем в окне набрать имя файла программы;
- из командной строки (Command Prompt): выбрать Start / Command Prompt, либо Start / Run / *cmd*; в окне командной строки набрать *имя программы* и нажать Enter.

В последних двух случаях *программа* должна входить в набор путей для поиска программ (*значение переменной окружения PATH*). Чтобы узнать или изменить *значение* этой *переменной окружения*, выберите Start / *Control Panel* / *System* / *Advanced System* Settings / *Environment Variables*. Получившееся окно позволит Вам наиболее удобным способом визуализировать или изменять значения переменных окружения.

Для управления Вашими задачами используйте программу *Windows* Task *Manager*, которую запустите, нажав одновременно клавиши Ctrl / *Alt* / *Del*. В результате визуализируется окно (рис. 35.9) :



увеличить изображение

Рис. 35.9. Окно программы Windows Task Manager

Вкладка Applications содержит информацию о вызванных Вами программах. В случае, например, зависания какой-либо программы, выберите ее и нажмите "*End* Task", в результате чего *программа* будет завершена. Вкладка Processes визуализирует информацию обо всех процессах, запущенных в системе. Вкладка *Performance* визуализирует информацию об использовании процессора и памяти, которая может оказаться Вам полезна в случае каких-либо незапланированных задержек в работе компьютера. Поэкспериментируйте с вкладками программы *Windows* Task *Manager*.

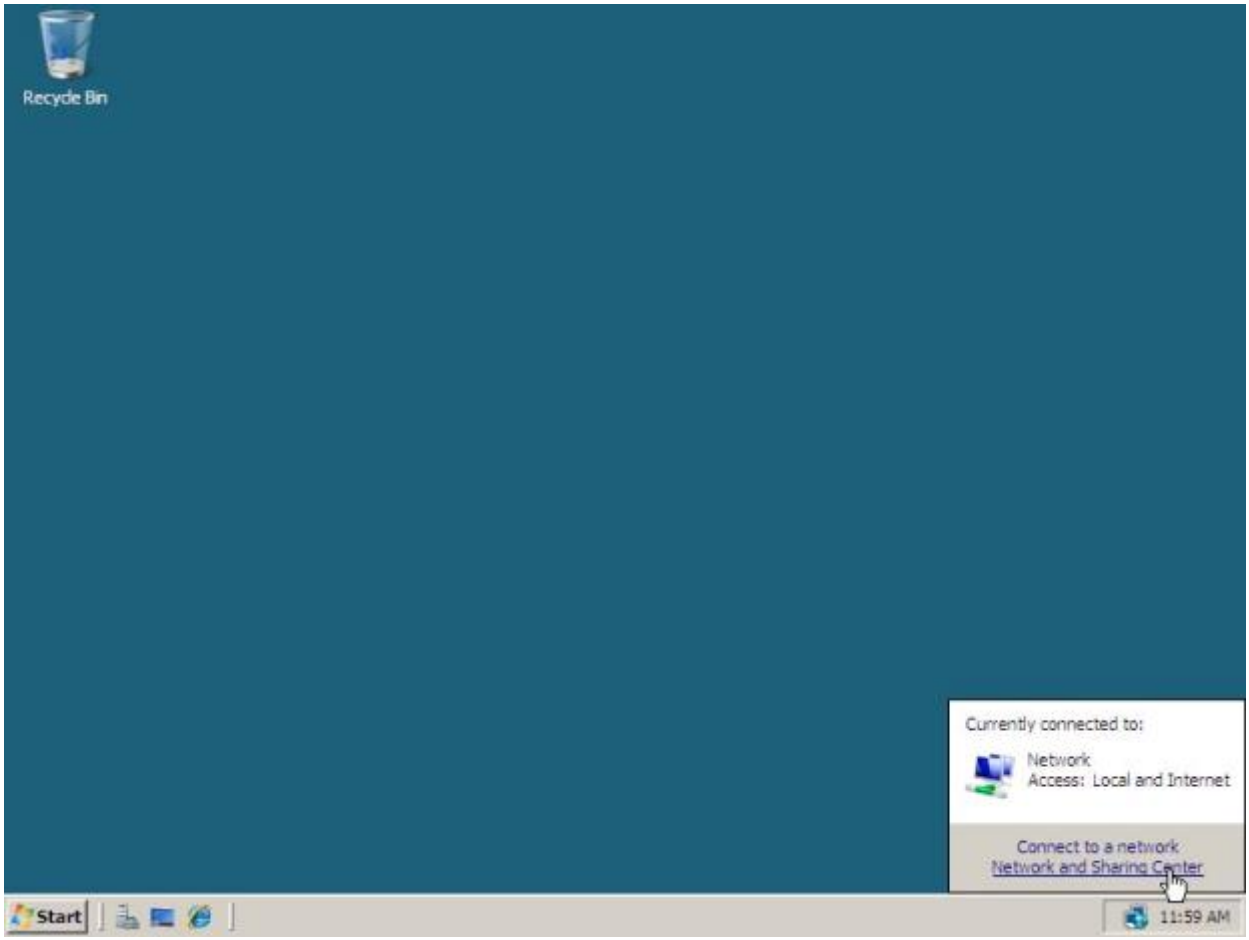
Сетевые установки

Для подсоединения компьютера к локальной *TCP/IP* - сети необходимо выполнить для него сетевые установки – задать IP-адрес и *сетевую маску*.

Физическое подсоединение к сети сделайте (проверьте) путем подсоединения к сетевому *разъему* (RJ45) сетевого кабеля вида twisted *pair* (*витая пара*), который соединяет Ваш *компьютер* с сетевым концентратором (*hub*) или переключателем (*switch*). Наличие физического соединения индицируется зеленым световым индикатором (проверьте).

Для соединения в *сеть* служит *сетевая карта* (сетевой *адаптер*). Ваша задача – правильно задать IP-адрес компьютера.

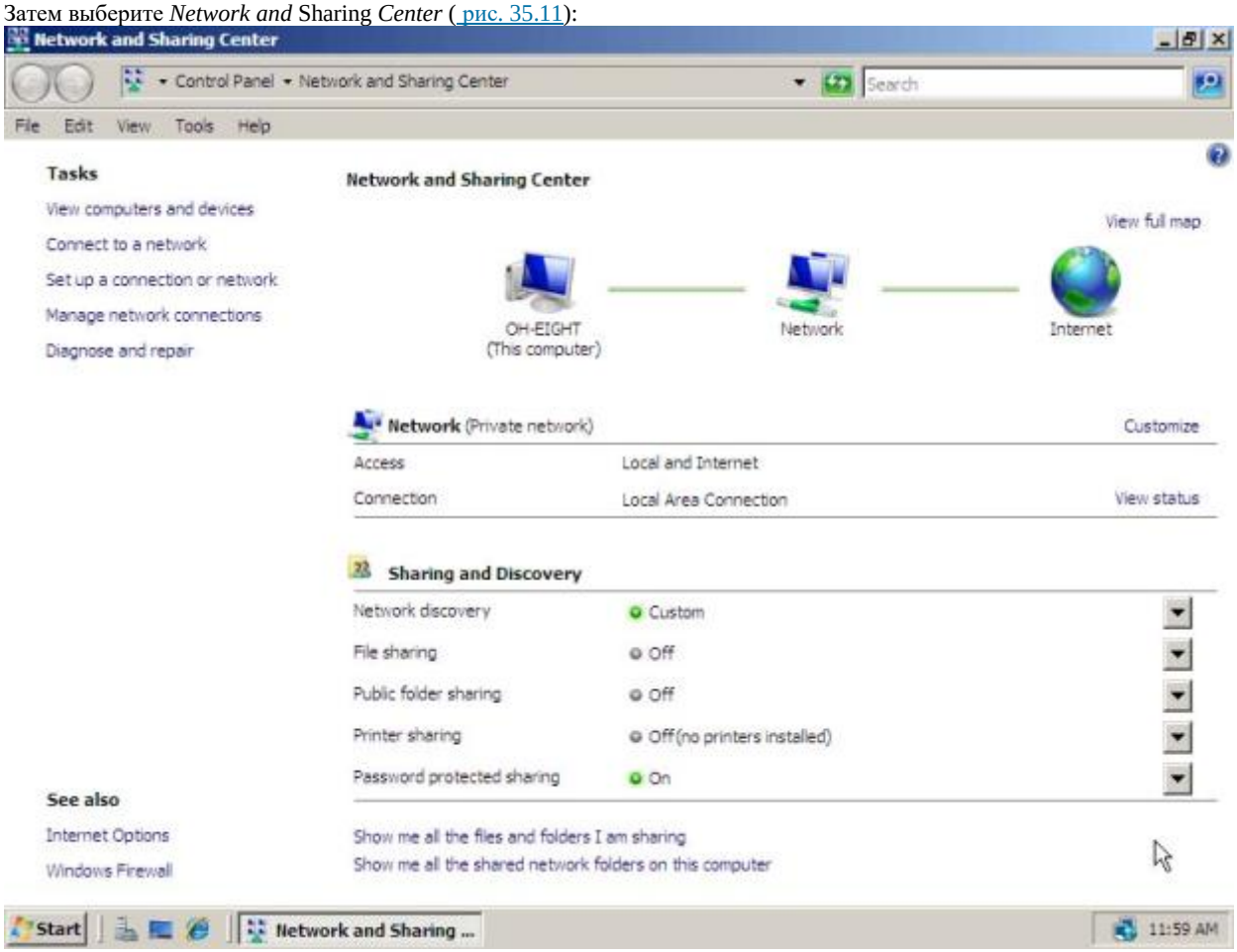
Для этого в стартовом *меню* выберите *пункт Network* (рис. 35.10):



увеличить изображение

Рис. 35.10. Выбор пункта Network: Переход к Network and Sharing Center

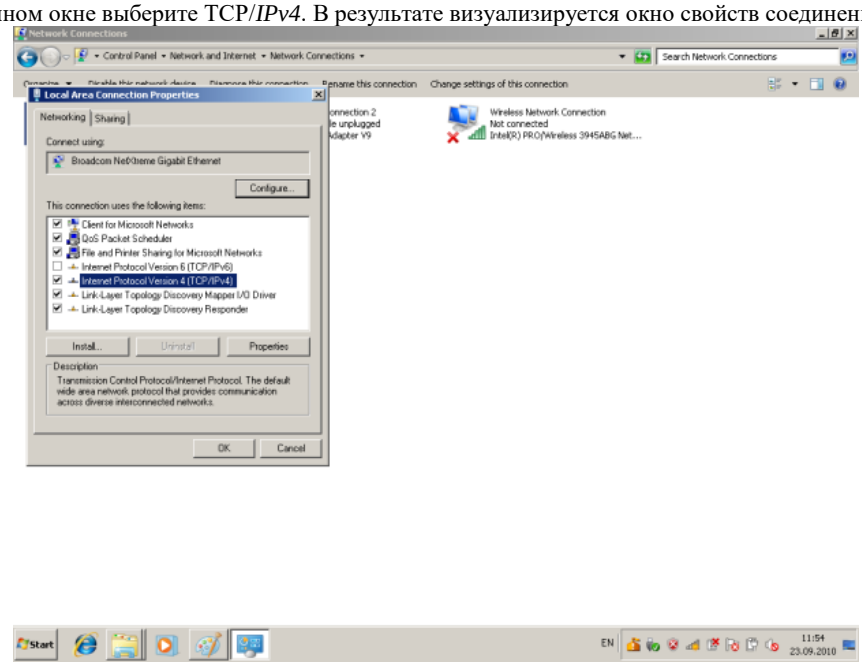
Затем выберите *Network and* Sharing *Center* (рис. 35.11):



увеличить изображение

Рис. 35.11. Network and Sharing Center

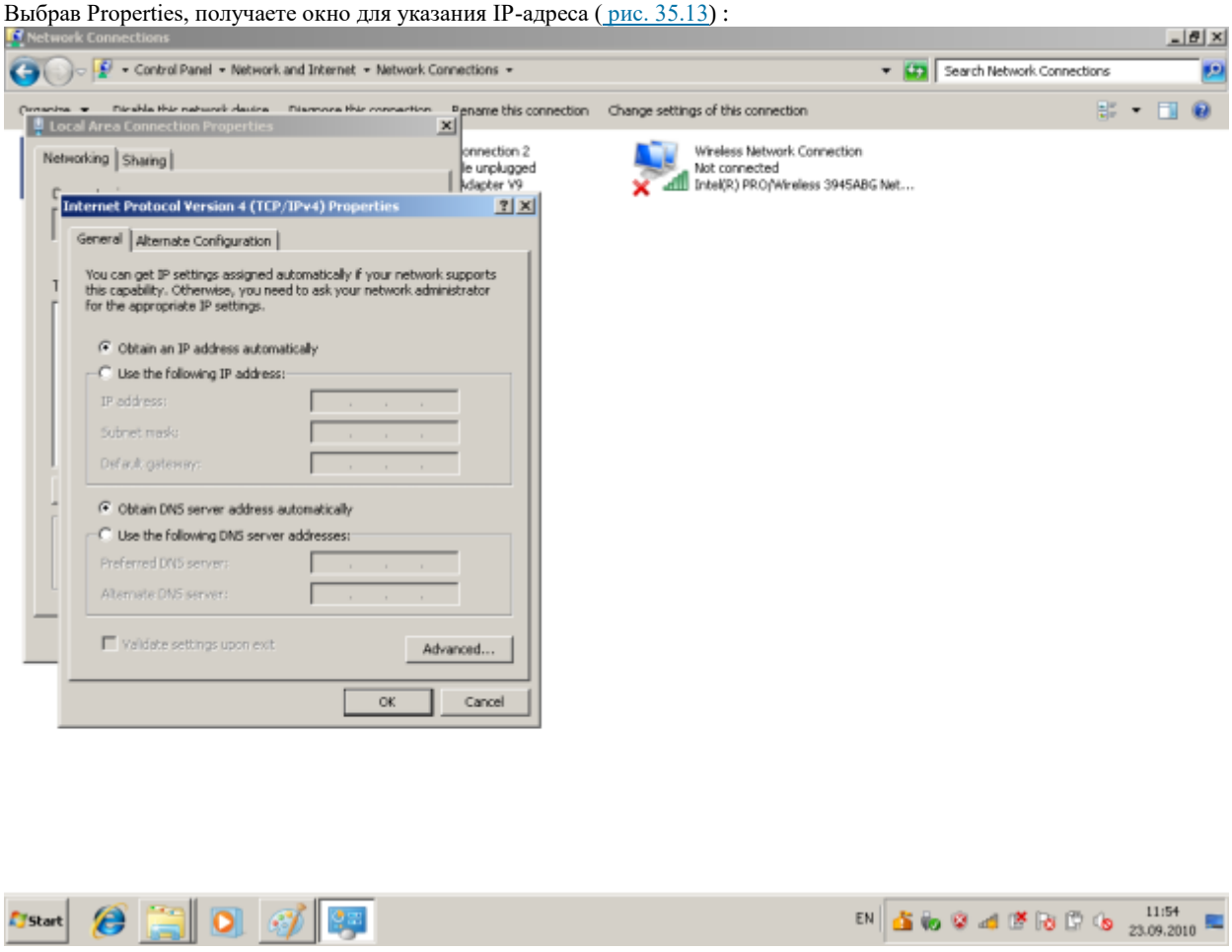
Выберите Manage *Network Connections* / *Local Area Connection* / Properties (свойства соединения по локальной сети). В полученном окне выберите TCP/*IPv4*. В результате визуализируется окно свойств соединения по локальной сети (рис. 35.12):



увеличить изображение

Рис. 35.12. Окно для выбора свойств сетевого TCP/IP – соединения

Выбрав Properties, получаете окно для указания IP-адреса (рис. 35.13) :



увеличить изображение

Рис. 35.13. Окно для указания IP-адреса

Как правило, по умолчанию выбран *пункт* "Obtain *IP address automatically*". Выберите *пункт* "Use the following *IP address*" и наберите IP-адрес Вашего компьютера и *сетевую маску* по образцу, показанному на рис. 35.13 . Нажмите OK. Система потребует от Вас перезапуска, чтобы изменения вступили в силу. Теперь Ваш *компьютер* готов к работе в локальной сети.

Работа на удаленных компьютерах

При работе в локальной сети очень полезная возможность *Windows* – удаленный вход на другой *компьютер* Вашей локальной сети. В *Windows* такая *функция* системы называется *Remote* Desktop *Connection* (удаленный *рабочий стол*). Для соединения Вы должны знать имя другого компьютера, например, aphrodite.

Для удаленного входа выберите Start / *All* Programs / *Accessories* / *Communications* / *Remote* Desktop *Connection*. В результате визуализуется окно (рис. 35.14):

Рис. 35.14. Окно удаленного рабочего стола

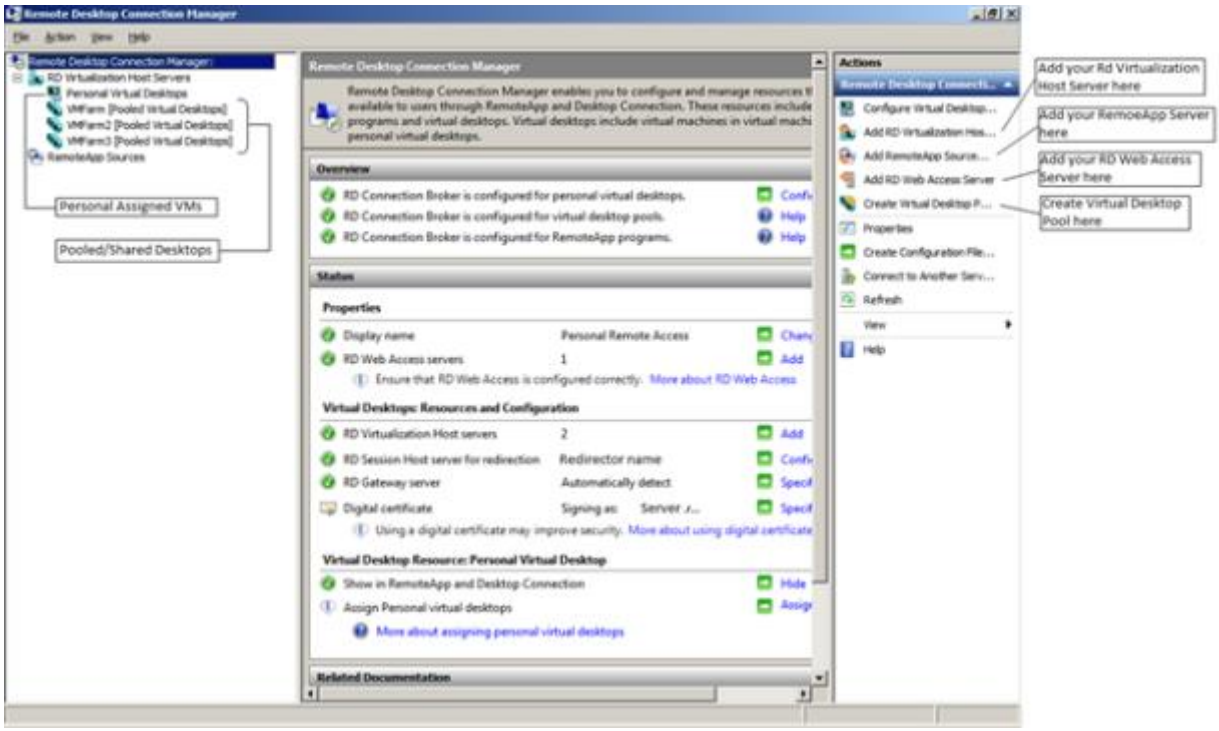
Наберите имя компьютера (или его IP-адрес, как показано на рисунке) и нажмите **Connect**.

Визуализируется окно входа на удаленный *компьютер*, в котором Вы должны набрать *имя пользователя* (логин) и *пароль*.

После этого экран Вашего компьютера используется как *терминал* для визуализации действий, выполняемых Вами на удаленном компьютере. Теперь выберите *Computer*, выведите имя компьютера и т.д., чтобы убедиться, что Вы теперь удаленно работаете на другом компьютере с указанным именем. *Компьютер* для удаленного входа выберите по указанию системного администратора локальной сети Вашего учебного класса.

Такая возможность очень удобна, если удаленный *компьютер* располагает необходимыми Вам ресурсами (памятью, быстрым процессором, инсталлированными на нем программами и др.), которых нет на Вашем компьютере.

Управление удаленным входом на другие компьютеры. Такая возможность введена в Windows 2008 Server R2:



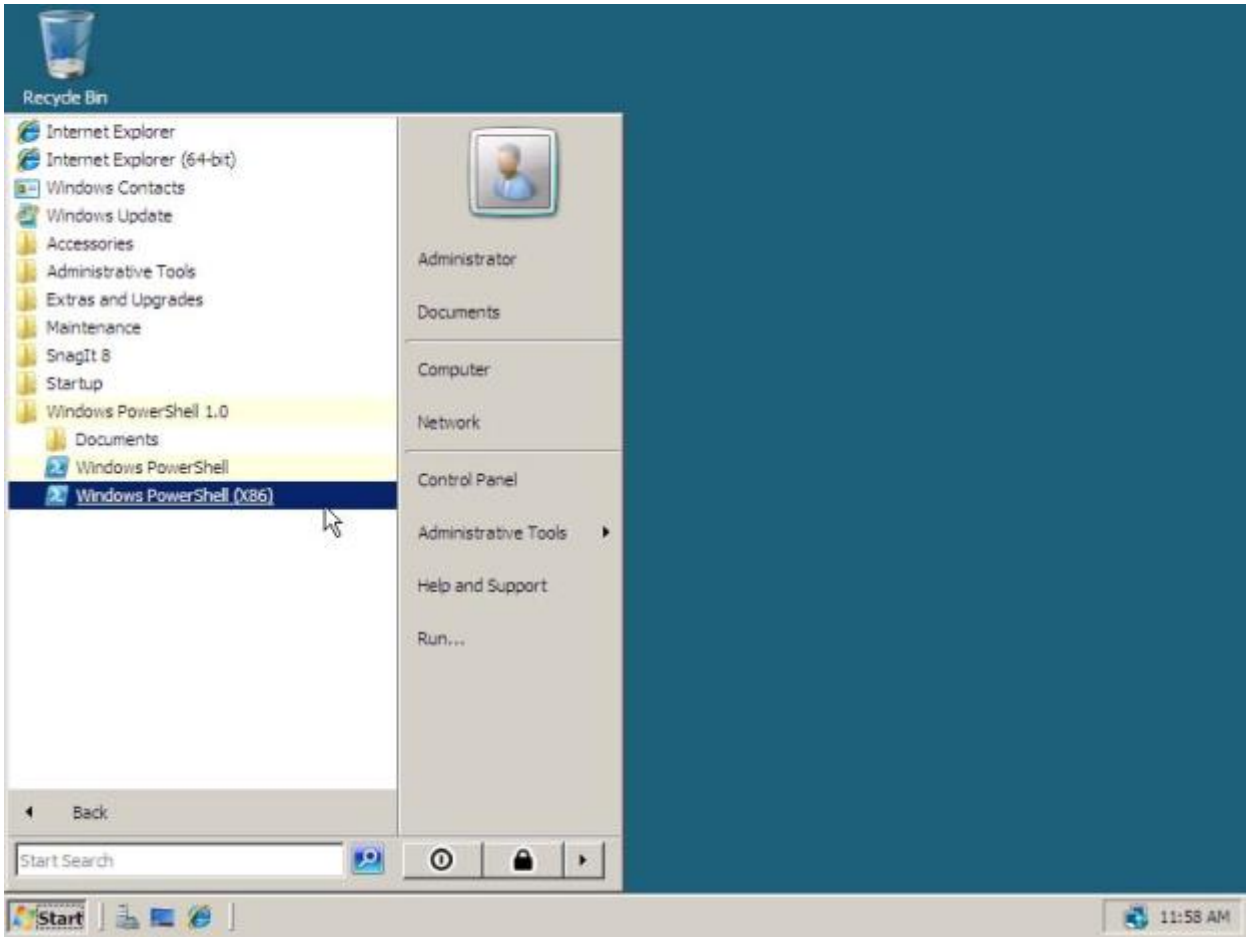
увеличить изображение

Рис. 35.15. Управление удаленным рабочим столом в Windows Server 2008 R2

Данная возможность позволяет Вам организовать несколько виртуальных рабочих столов для работы на нескольких компьютерах и управлять ими.

Командный процессор PowerShell

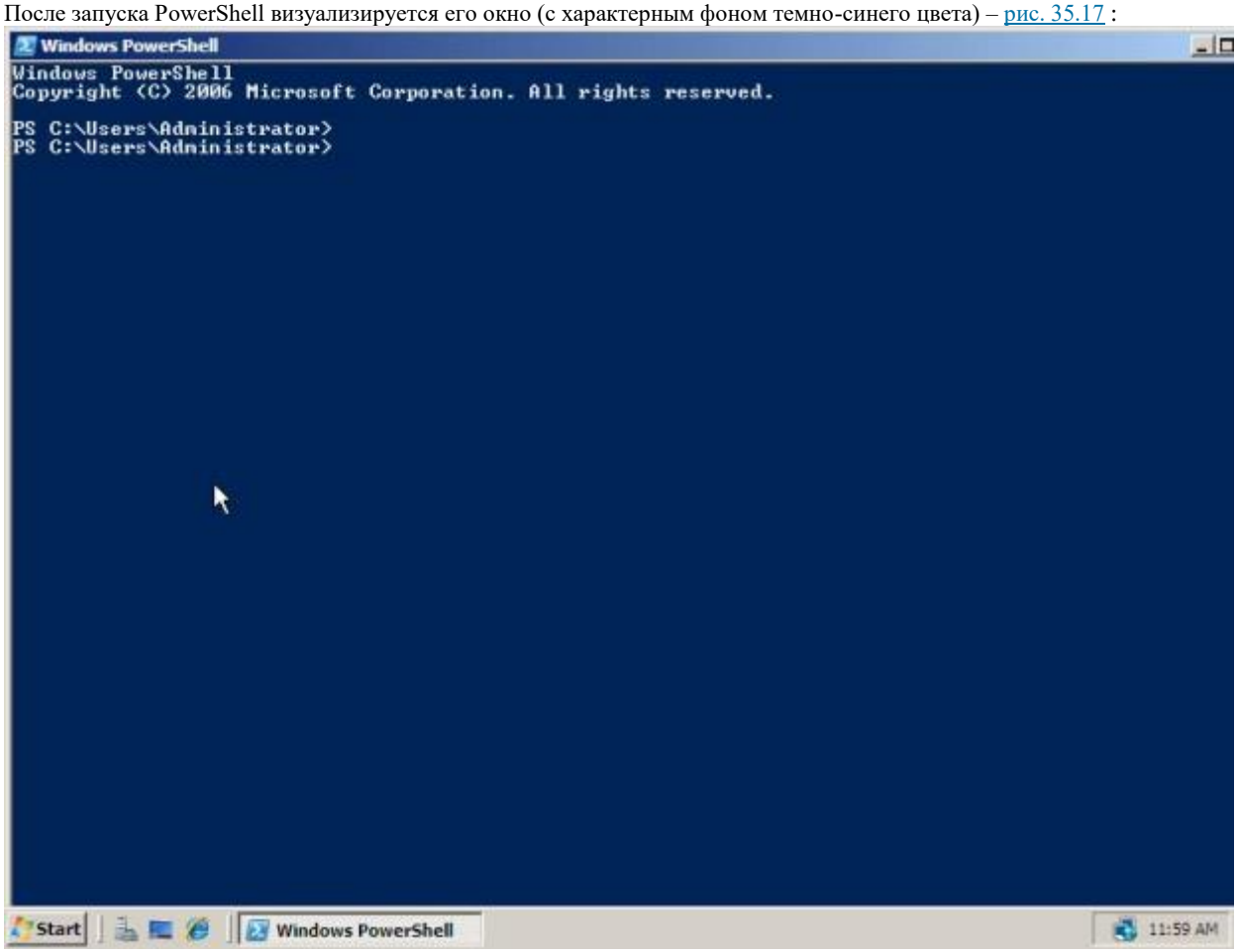
Командный процессор PowerShell (новая возможность *Windows* 2008) запускается из пункта *меню* Start / *All* Programs / *Windows* PowerShell *x86* (x64) – рис. 35.16



увеличить изображение

Рис. 35.16. Запуск PowerShell

После запуска PowerShell визуализируется его окно (с характерным фоном темно-синего цвета) – рис. 35.17 :



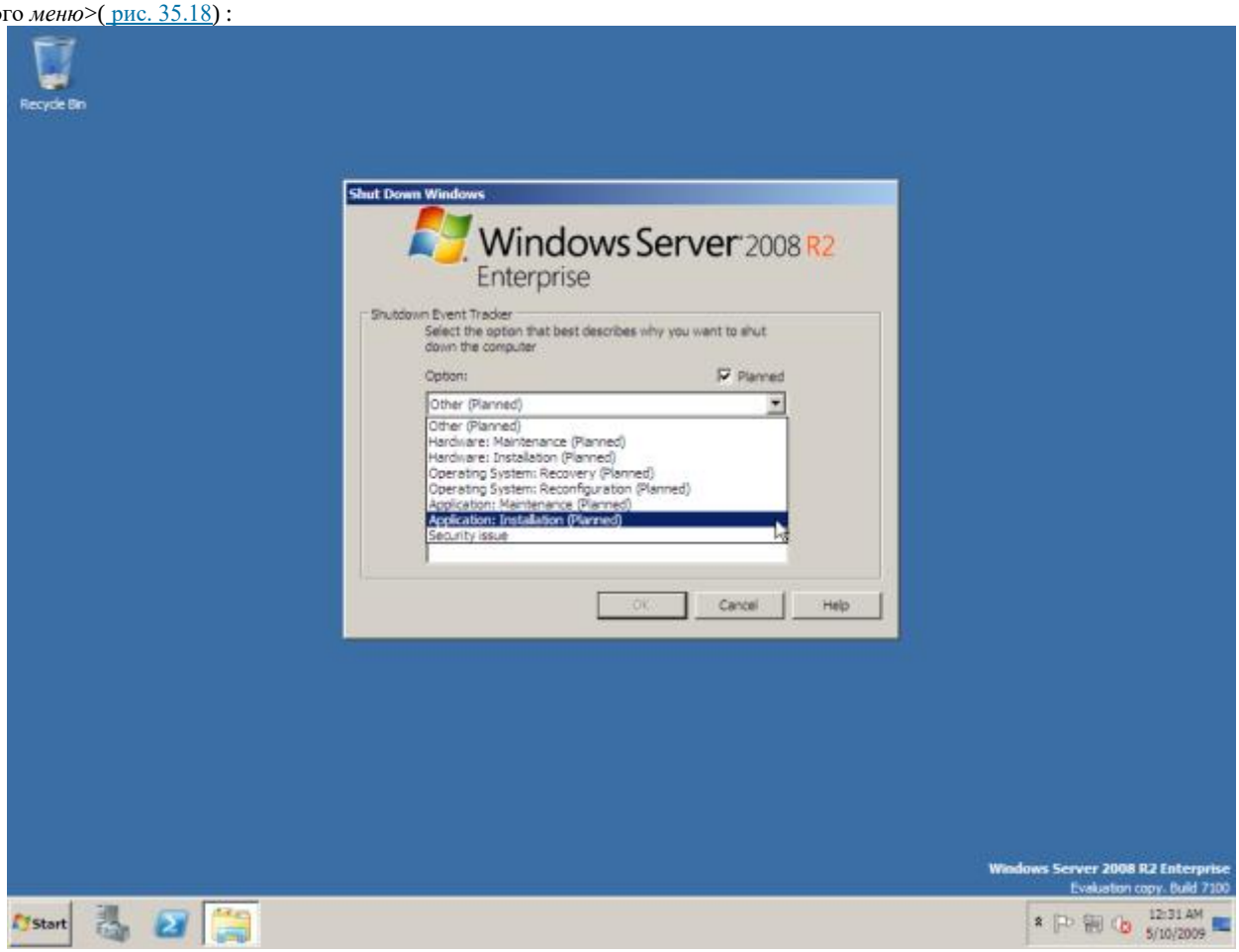
увеличить изображение

Рис. 35.17. Командный процессор PowerShell вызван: Набирайте команды

С командами и другими возможностями PowerShell ознакомьтесь в справке (*Help*) по *Windows* 2008 и поэкспериментируйте с ним.

Выход из системы

Для выхода из Вашего сеанса пользователя выберите Start / <кнопка выключения питания в нижней части стартового *меню*>(рис. 35.18) :



увеличить изображение

Рис. 35.18. Выключение компьютера или выход из системы

В результате произойдет *выход* из Вашего сеанса пользователя, затем – выгрузка ОС и *выключение компьютера*. Перед этим *Windows* 2008 (как *серверная ОС*) предложит Вам более детально пояснить причину остановки или перезапуска сервера.

В данной лабораторной работе Вы познакомились лишь с некоторыми базовыми возможностями ОС *Windows* 2008. Более подробно с ними можно познакомиться в книге [7], а с внутренней организацией и архитектурой системы – в книге [14].

Для более детального ознакомления с возможностями *параллельных вычислений* и реализации параллельных алгоритмов решения задач прикладной математики для *Windows* 2008 рекомендуем Вам ознакомиться с результатами и исходными кодами нашего проекта Palallel Dwarfs [15], выполненным совместно СПбГУ (под научным руководством проф. В.О. Сафонова) и Microsoft в среде *Windows* 2008 / *Visual Studio* 2010 на языках C++, C# и новом функциональном языке Microsoft – F#. *Сайт* проекта содержит исходные коды и документацию, включающую руководство по инсталляции и запуску системы, а также пользовательский форум для обсуждений и систему для фиксации и исправления ошибок. Вы можете стать участником этого открытого проекта, войдя на *сайт*, зарегистрировавшись и участвуя в разработке и обсуждениях.

Контрольные вопросы ПЗ2 (ПК-1):

1. Что такое системный вызов?(ПК-1).
2. Какими способами могут передаваться параметры системному вызову?(ПК-1).
3. Какие виды системных вызовов Вы знаете?(ПК-1).
4. Как организовано распределение памяти в MS DOS?(ПК-1).
5. Как организовано распределение памяти для нескольких задач в UNIX?(ПК-1).
6. Каковы способы реализации коммуникационных моделей взаимодействия между процессами?(ПК-1,).
7. Что такое системные программы и какие функции они выполняют?(ПК-1,).
8. Какова архитектура MS DOS?(ПК-1,).

9. Какова архитектура UNIX?(ПК-1,).
10. Что такое уровни абстракции и каким образом облегчается разработка ОС на основе уровней абстракции?(ПК-1,).
11. Что такое уровень абстракции и какие ограничения накладываются на реализацию его операций?(, ПК-1).
12. Какие уровни абстракции реализованы в системе OS/2?(, ПК-1).
13. Что такое микроядро и как организуются операционные системы по принципу микроядра?(, ПК-1).
14. В чем преимущество разработки ОС по принципу микроядра?(, ПК-1).
15. По каким принципам организована ОС Windows NT и с приложениями для каких платформ она поддерживает совместимость?(, ПК-1).
16. Что такое виртуальная машина и каким образом концепция виртуальной машины используется при разработке ОС?(, ПК-1).
17. В чем преимущества для пользователя при работе в персональной виртуальной машине в рамках операционной системы?(ПК-1).
18. Что такое виртуальная машина Java (JVM) и из каких компонент она состоит?(ПК-1).
19. Что такое загрузчик классов в JVM?(ПК-1).
20. Что такое верификатор в JVM?(ПК-1).
21. Что такое интерпретатор в JVM?(ПК-1).
22. Что такое JIT-компилятор в JVM?(ПК-1).

ПРАКТИЧЕСКАЯ РАБОТА № 3.

Изучение системы Windows

Целью лабораторной работы является практическое освоение операционной системы Windows XP – ее графической оболочки, входа и выхода, структуры рабочего стола, основных действий и настроек при работе в системе. Необходимый общий теоретический материал по архитектуре и особенностям ОС Windows представлен в "Обзор архитектуры и возможностей систем Windows 2000/XP/2003/Vista/2008/7 " и "Системные механизмы Windows " данного курса.

Содержание

Аппаратура и программные инструменты, необходимые для лабораторной работы

Настольный или портативный компьютер с операционной системой Microsoft Windows XP

Продолжительность лабораторной работы

2 академических часа

Обзор Windows XP

Windows XP (от **eXPerience** – опыт) – до сих пор (2010 г.) наиболее широко используемая в мире клиентская операционная система (доля рынка – 53% на август 2010 г.) фирмы Microsoft. Система выпущена в 2001 г. К ней выпущены три сервис-пака – SP1, SP2, SP3. Рекомендуется использование данной ОС с инсталлированным третьим сервис-паком (SP3).

Система доступна в нескольких версиях. Наиболее полная – Windows XP Professional.

Запуск системы

Включите компьютер с инсталлированной Windows XP + SP3.

Через 0.5 – 1 мин. (примерное время загрузки системы) на экране появится баннер "Microsoft Windows XP", затем – стартовая страница для входа с именами пользователей (рис. 33.1).

Вход в систему и аутентификация пользователя

Выберите Ваше имя пользователя и кликните мышкой по картинке рядом с именем. Как правило, в систему уже введено стандартное имя User. Если для пользователя установлен пароль, введите его.

Если в системе включен звук, будет проиграна характерная для данной версии Windows короткая мелодия на фортепиано из шести нот, из-за которой, по-видимому, система при разработке получила кодовое название Whistler (свистулька).

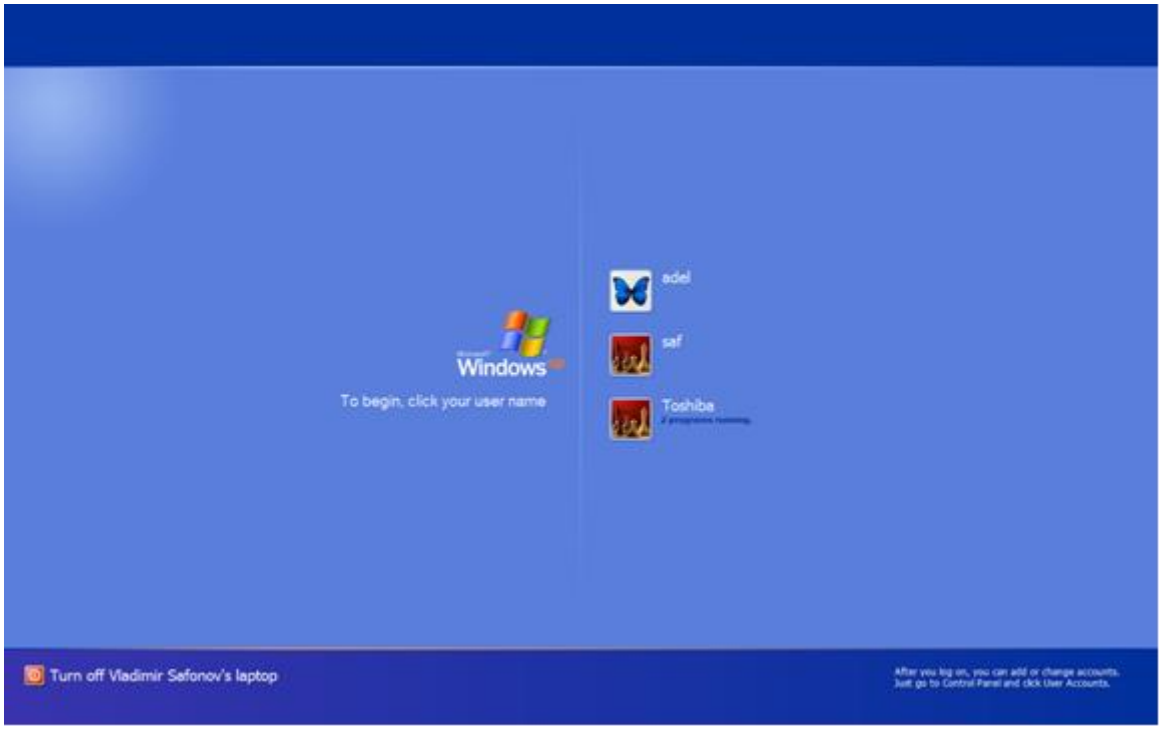


Рис. 33.1. Стартовое меню для входа пользователей в систему

После входа в систему на экране визуализируется рабочий стол (desktop) (рис. 33.2) :

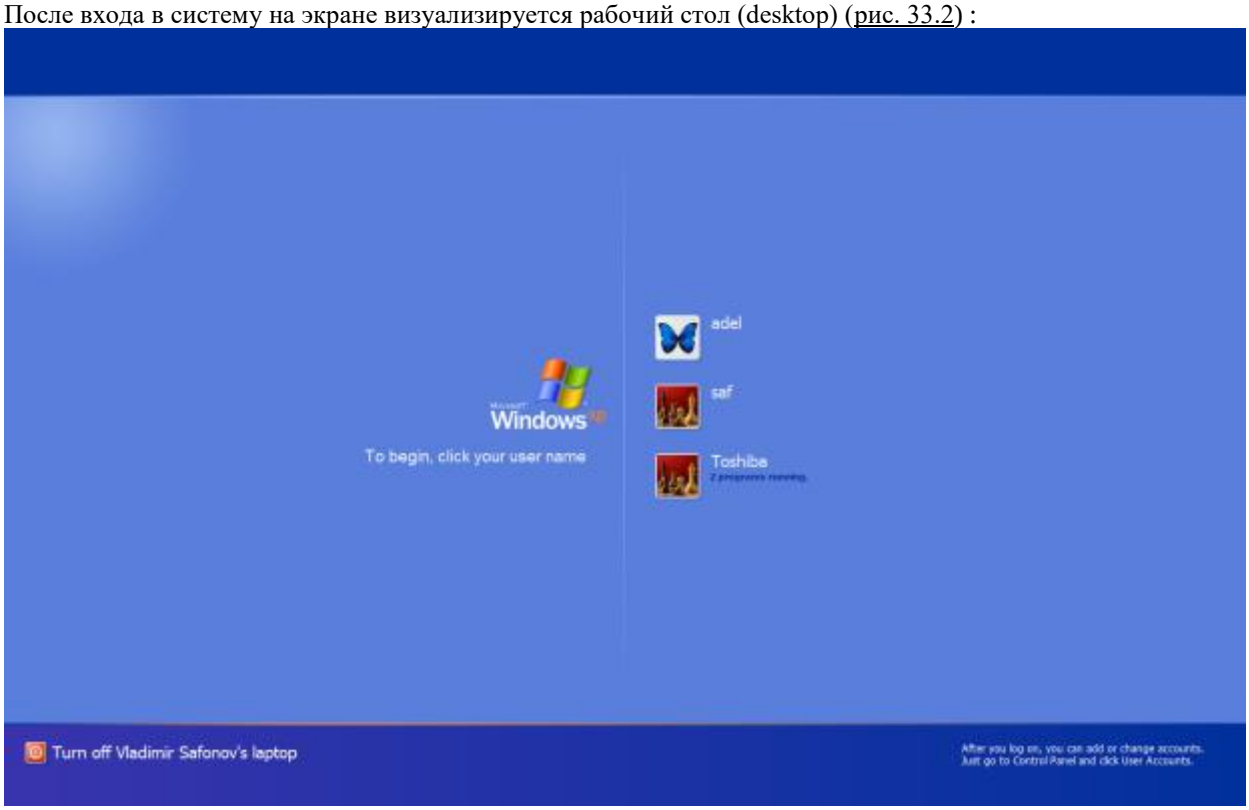


Рис. 33.2. Рабочий стол Windows XP

Структура рабочего стола, мой компьютер, панель управления

Рабочий стол состоит из иконок приложений (например, Internet Explorer) и панели задач (taskbar) – обычно синего цвета, в нижней части. В левом нижнем углу расположена кнопка Start, при нажатии на которую пользователь может выбрать начальное действие – запуск какого-либо приложения, создание документа и др. (рис. 33.3).

Вид и фон рабочего стола при разных настройках могут отличаться. На рис. 33.2 показан один из типичных для Windows XP фонов рабочего стола – Bliss (букв. "блаженство"). Для изменения фона рабочего стола необходимо на фоновом рисунке нажать правую кнопку мыши и в контекстном меню выбрать Properties / Desktop, после чего выбрать нужный рисунок фона в выпадающем списке.

Рис. 33.3. Состояние рабочего стола после нажатия кнопки Start

Основные пункты стартового меню, визуализируемого в результате нажатия кнопки Start:

- My Computer – информация о компьютере, его ресурсах, устройствах, имени, установеленной на нем ОС
- My Documents – стандартная папка для создаваемых документов (Вы можете помещать документы и в любую другую более удобную Вам папку)
- My Network Places – узлы локальной сети, доступные с компьютера
- Control Panel – панель управления (№%i0004)
- (в нижней части) Log Off – выход из Вашего пользовательского сеанса
- (в нижней части) Turn Off Computer – выключение компьютера или перезапуск системы.

Рис. 33.4. Панель управления

Рассмотрите более подробно панель управления (рис. 33.4). Она позволяет управлять ресурсами компьютера. Например, пункт Add or Remove Programs позволяет инсталлировать новые программы, деинсталлировать или инсталлировать вновь ("ремонтировать") уже установленные.

Выберите в стартовом меню пункта My Computer, При этом в специальном окне визуализируется информация о состоянии компьютера (рис. 33.5):

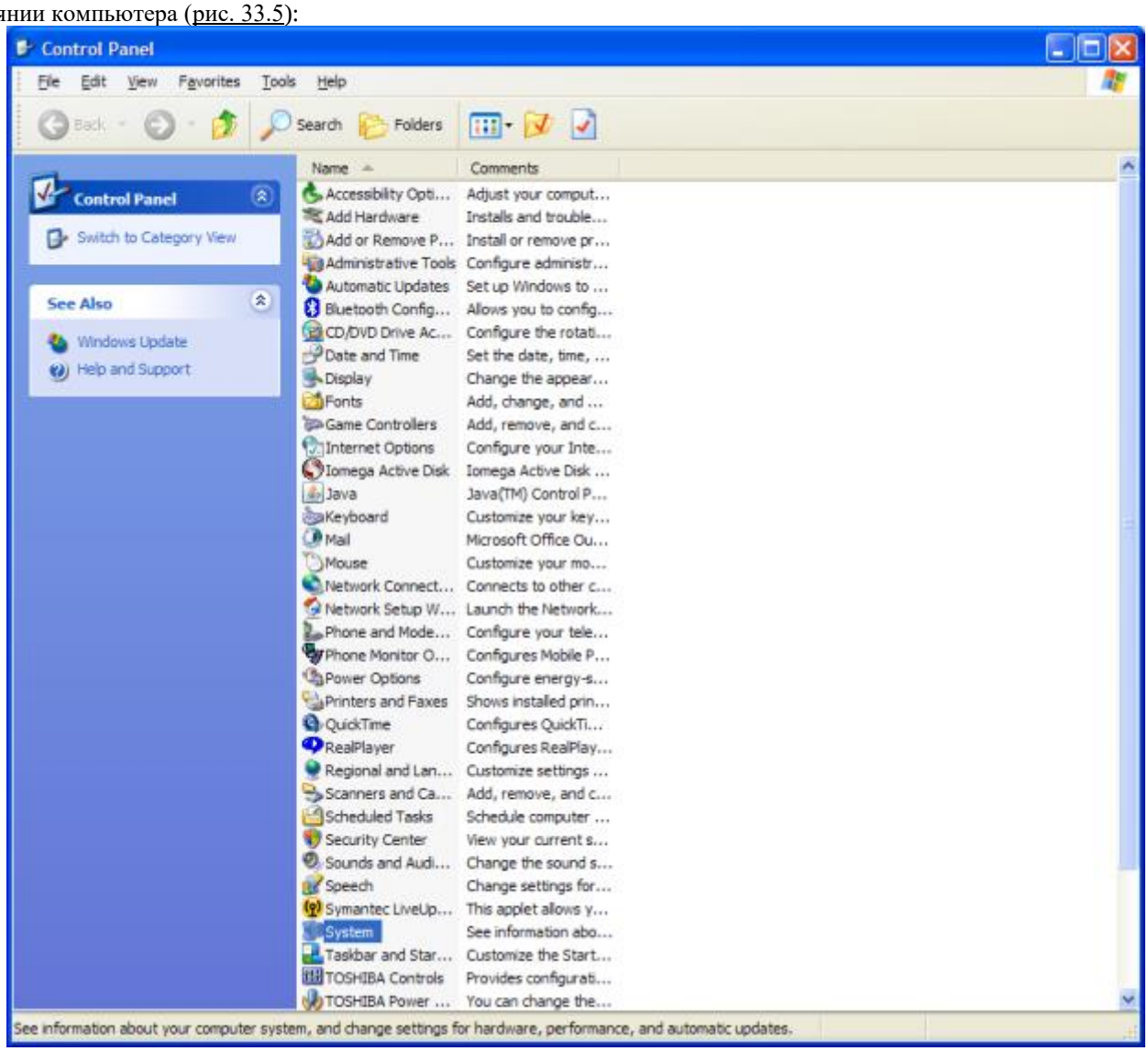


Рис. 33.5. My Computer

В окне My Computer (рис. 33.5) визуализируется информация о дисках и некоторых наиболее важных папках и предлагается набор возможных действий и набор других информационных узлов для перехода к ним (например, My Network Places).

Для визуализации основных свойств компьютера (системной информации) выберите в стартовом меню: My Computer / (Правая кнопка мыши) / Properties. Возникает окно с системной информацией (рис. 33.6):

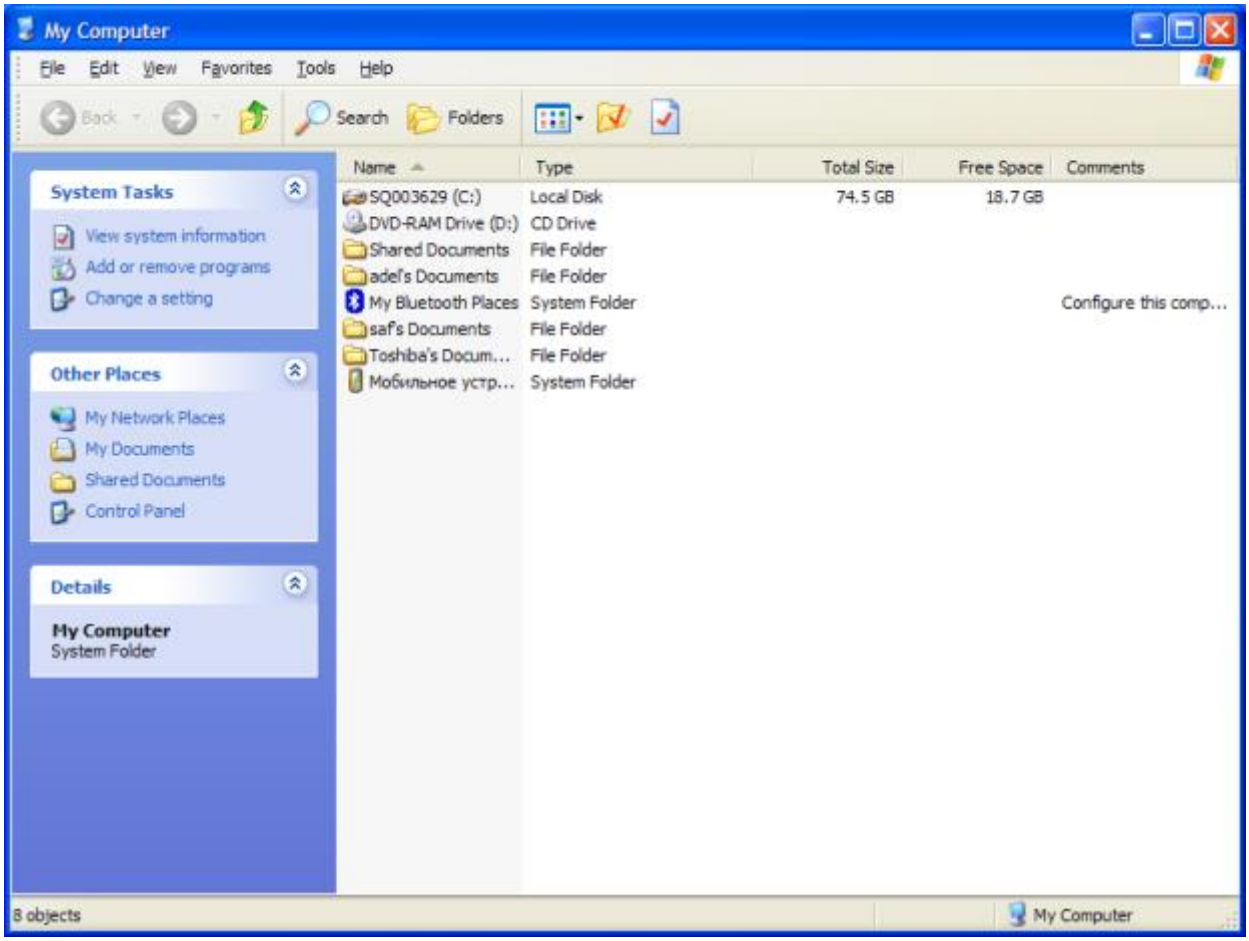


Рис. 33.6. Системная информация о компьютере

Вы видите информацию об ОС, объеме памяти, типе процессора и ряд вкладок, например, Computer Name, кликнув на которой, получаете информацию об имени компьютера. Кликнув на вкладку Hardware / Device Manager, получите подробную информацию о составе оборудования компьютера и установленных драйверах

Работа с файлами и папками

Работа с файлами и папками (folders) – хранилищами ссылок на файлы и другие папки – осуществляется с помощью программы Windows Explorer. На рис. 33.5 показано окно программы Windows Explorer, визуалилизующее информацию о дисках и основных папках компьютера. Если дважды подряд кликнуть на диске C:, то визуализируется содержимое его корневой папки, которая может содержать другие папки, и т.д. С помощью зеленой стрелки (Up) можно вернуться вверх на родительскую директорию.

Выбор файла или папки в директории осуществляется одним кликом мышки, вход в директорию или открытие файла – двойным кликом мышки на имени директории или файла. При этом для файла выполняется действие его открытия, зависящее от его типа, - для текстовых файлов – вызов соответствующего редактора (notepad, WordPad, MS Word и др.), для файлов .pdf – вызов Adobe Acrobat, для исполняемых кодов или командных файлов – запуск соответствующей программы или скрипта и т.д. Поэкспериментируйте на своем компьютере с навигацией по файлам и папкам и открытием файлов с документами.

Запуск программ, управление задачами, программами и процессами

Есть несколько способов запустить программу:

- из Windows Explorer – дважды кликнуть на имени ее файла;
- из пункта меню Start – выбрать пункт Run, затем в окне набрать имя файла программы (без расширения .exe); например, notepad – при этом вызовется стандартный редактор текстовых (ASCII) файлов;
- из командной строки (Command Prompt): выбрать Start / All Programs / Accessories / Command Prompt, либо Start / Run / cmd; в окне командной строки набрать имя программы и нажать Enter.

В последних двух случаях программа должна входить в набор путей для поиска программ (значение переменной окружения PATH). Чтобы узнать или изменить значение этой переменной окружения, выберите Start / My Computer / (правая кнопка мыши) Properties / Advanced / Environment Variables. Получившееся окно позволит Вам наиболее удобным способом визуализировать или изменять значения переменных окружения.

Для управления Вашими задачами используйте программу Windows Task Manager, которую запустите, нажав и 1-2 секунды подержав одновременно клавиши Ctrl / Alt / Del. В результате визуализируется окно (рис. 33.7):

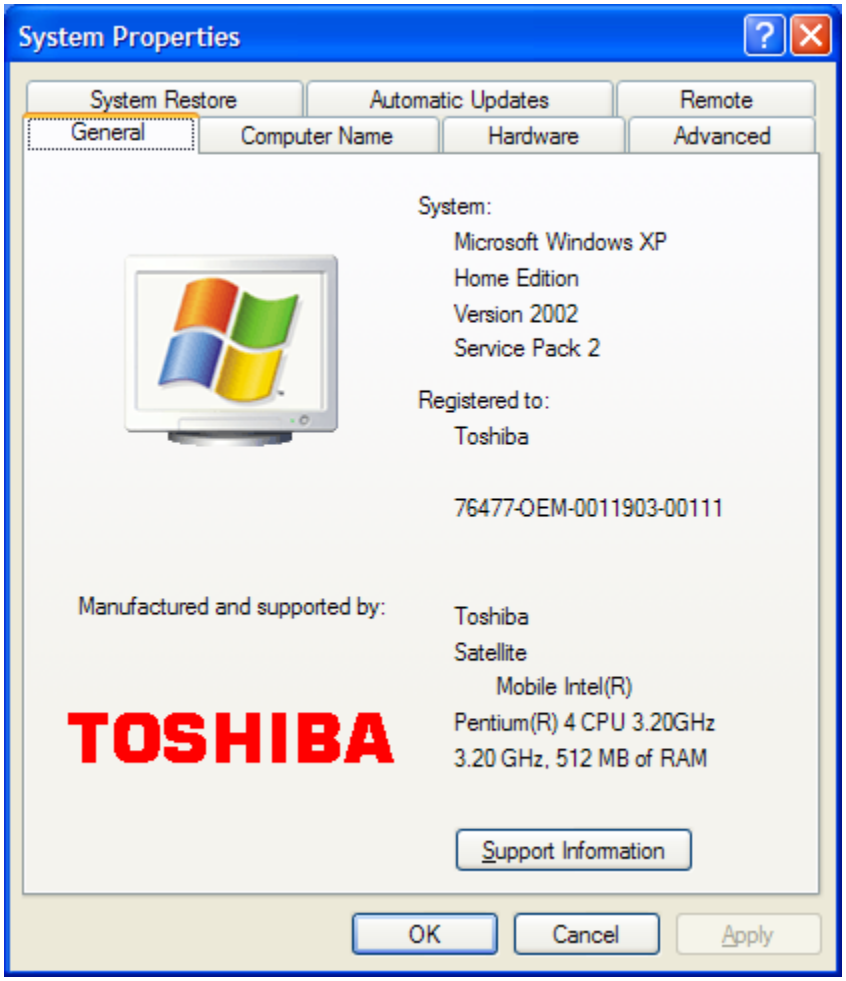


Рис. 33.7. Окно программы Windows Task Manager

Вкладка Applications содержит информацию о вызванных Вами программах. В случае, например, зависания какой-либо программы, выберите ее и нажмите "End Task", в результате чего программа будет удалена из системы. Вкладка Processes визуализирует информацию обо всех процессах, запущенных в системе. Вкладка Performance визуализирует информацию об использовании процессора и памяти, которая может оказаться Вам полезна в случае каких-либо незапланированных задержек в работе компьютера. Поэкспериментируйте со вкладками программы Windows Task Manager.

Сетевые установки

Для подсоединения компьютера к локальной TCP/IP - сети необходимо выполнить для него сетевые установки – задать IP-адрес и сетевую маску.

Физическое подсоединение к сети сделайте (проверьте) путем подсоединения к сетевому разъему (RJ45) сетевого кабеля вида twisted pair (витая пара), который соединяет Ваш компьютер с сетевым концентратором (hub) или переключателем (switch). Наличие физического соединения индицируется зеленым световым индикатором (проверьте).

Для соединения в сеть служит сетевая карта (сетевой адаптер). Ваша задача – правильно задать IP-адрес компьютера. Для этого выберите Start / Connect To / Show All Connections. В визуализируемом окне показаны все сетевые соединения (Network Connections) Вашего компьютера. Выберите из них соединение по локальной сети (Local Area Connection) и, нажав правую кнопку мышки, выберите Properties. В окне Local Area Connection Properties, в списке протоколов выберите Internet Protocol (TCP/IP) (рис. 33.8):

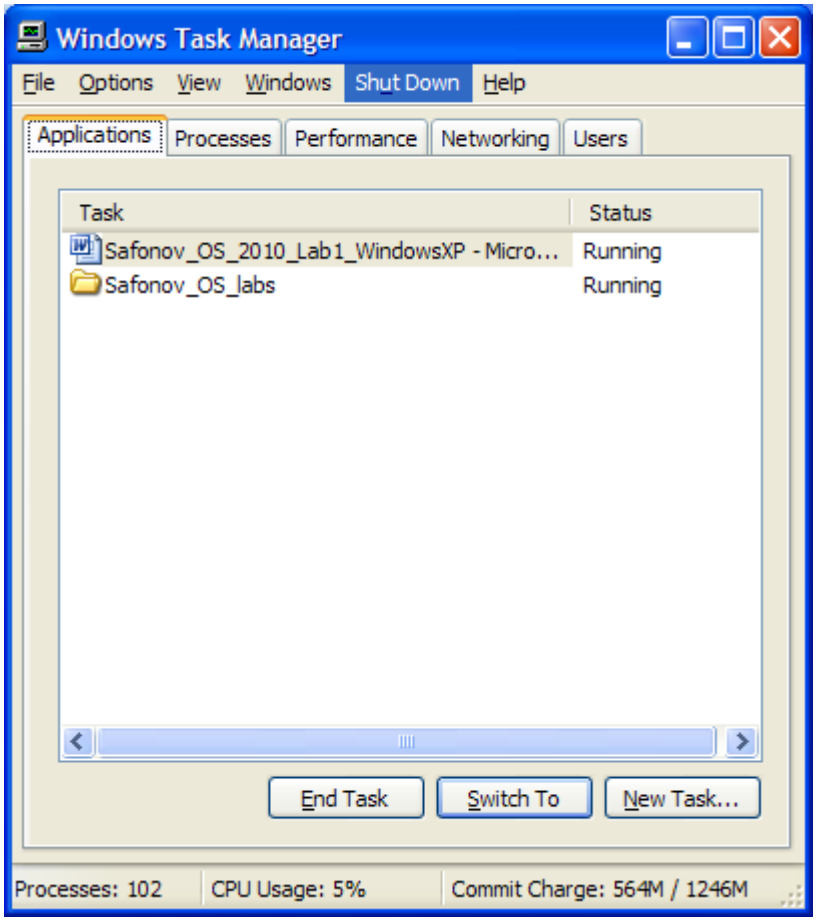


Рис. 33.8. Окно свойств сетевого соединения по локальной сети.

Затем нажмите Properties (т.е. свойства TCP/IP – соединения). Визуализируется окно, изображенное на рис. 33.9:

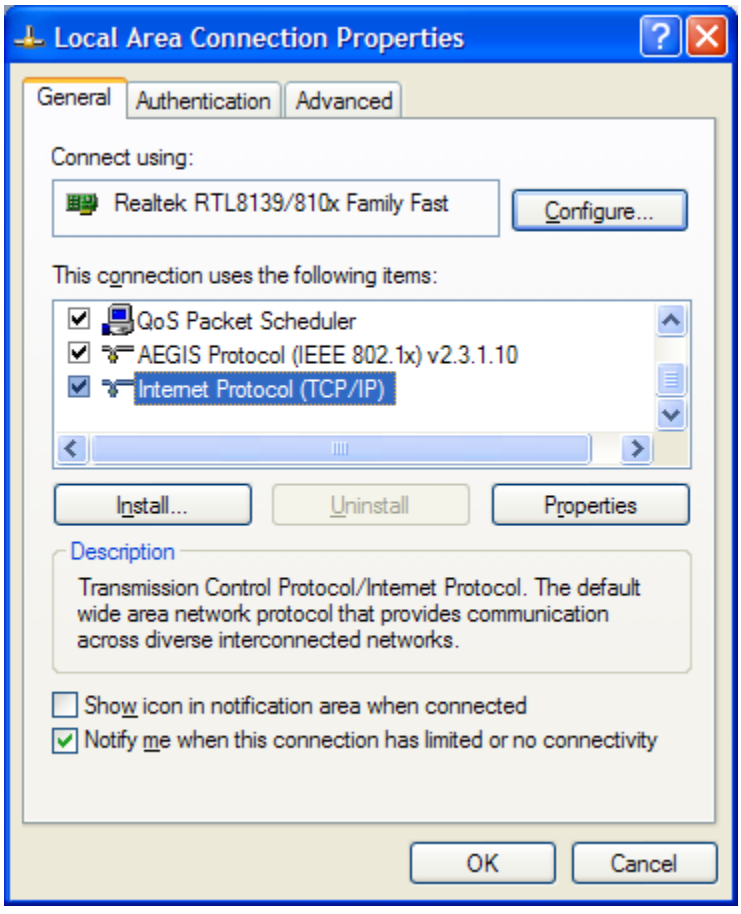


Рис. 33.9. Окно свойств сетевого TCP/IP – соединения

Как правило, по умолчанию выбран пункт "Obtain IP address automatically". Выберите пункт "Use the following IP address" и наберите IP-адрес Вашего компьютера и сетевую маску по образцу, показанному на рис. 33.9 . Нажмите OK. Система потребует от Вас перезапуска, чтобы изменения вступили в силу. Теперь Ваш компьютер готов к работе в локальной сети.

Работа на удаленных компьютерах

При работе в локальной сети очень полезная возможность Windows XP – удаленный вход на другой компьютер Вашей локальной сети. В Windows такая функция системы называется Remote Desktop Connection (удаленный рабочий стол). Для соединения Вы должны знать имя другого компьютера, например, **aphrodite**.

Для удаленного входа выберите Start / All Programs / Accessories / Communications / Remote Desktop Connection. В результате визуалилизуется окно (рис. 33.10):

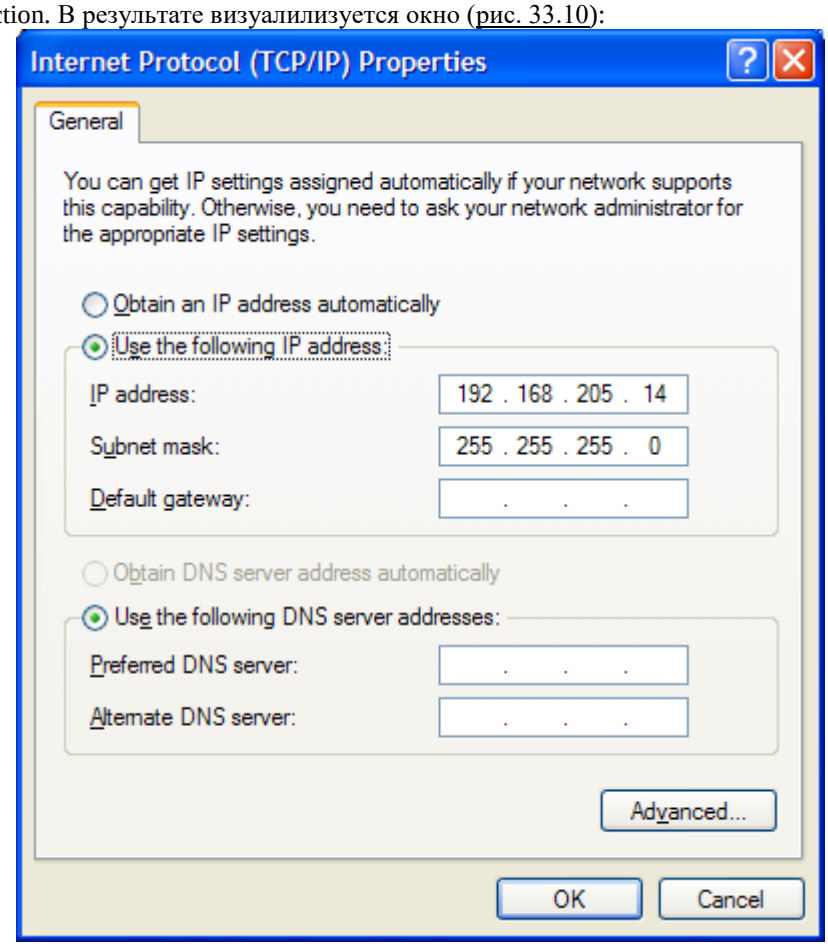


Рис. 33.10. Окно удаленного рабочего стола

Наберите имя компьютера и нажмите **Connect**.

Визуализируется окно входа на удаленный компьютер, в котором Вы должны набрать имя пользователя (логин) и пароль.

После этого экран Вашего компьютера используется как терминал для визуализации действий, выполняемых Вами на удаленном компьютере. Теперь выберите My Computer, выведите имя компьютера и т.д., чтобы убедиться, что Вы теперь удаленно работаете на другом компьютере с указанным именем. Компьютер для удаленного входа выберите по указанию системного администратора локальной сети Вашего учебного класса.

Такая возможность очень удобна, если удаленный компьютер располагает необходимыми Вам ресурсами (памятью, быстрым процессором, инсталлированными на нем программами и др.), которых нет на Вашем компьютере.

Выход из системы

Для выхода из Вашего сеанса пользователя выберите Start / Log Off / Log Off.

Визуализируется стартовое меню для входа пользователя в систему (рис. 33.1).

Для выхода с выключением компьютера выберите Start / Turn Off Computer / Turn Off (после выбора Start нажмите две подряд красных кнопки).

В результате произойдет выход из Вашего сеанса пользователя, затем – выгрузка ОС и выключение компьютера. При выходе из системы проигрывается характерная для Windows XP заключительная фортепианная мелодия из четырех нот.

В данной лабораторной работе Вы познакомились лишь с некоторыми базовыми возможностями ОС Windows XP. Более подробно с ними можно познакомиться в книге [12] , а с внутренней организацией и архитектурой системы – в книге [7] .

3.3. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните принципы построения суммирующего и вычитающего счётчиков по табл. 1 и 2.
2. Как реализуется параллельное формирование сигнала переноса во всех разрядах счётчика?
3. Поясните построение и работу реверсивного счётчика.
4. Как функционируют выводы “>15” и “<0” реверсивного счётчика, каково их практическое применение?

Контрольные вопросы ПЗ3 (ПК-4):

1. Какие операции определяет понятие «обращение к ЗУ»?
2. Какие единицы измерения используются для указания емкости запоминающих устройств?
3. В чем отличие между временем доступа и периодом обращения к запоминающему устройству?
4. Чем вызвана необходимость построения системы памяти по иерархическому принципу?
5. Что включает в себя понятие «локальность по обращению»?
6. Благодаря чему среднее время доступа в иерархической системе памяти определяется быстродействующими видами ЗУ?
7. Что в иерархической системе памяти определяют термины «промах» и «попадание»?
8. На какие вопросы необходимо ответить, чтобы охарактеризовать определенный уровень иерархической памяти?
9. Какие виды запоминающих устройств может содержать основная память?
10. Охарактеризуйте возможные варианты построения блочной памяти.
11. Какие возможности по сокращению времени доступа к информации предоставляет блочная организация памяти?
12. Чем обусловлена эффективность расслоения памяти?
13. Какая топология запоминающих элементов лежит в основе организации полупроводниковых ЗУ?
14. Какое минимальное количество линий должен содержать столбец ИМС памяти?
15. Поясните назначение управляющих сигналов в микросхеме памяти.
16. Чем отличаются страничный, быстрый страничный и пакетный режимы доступа к памяти?
17. Чем обусловлена необходимость регенерации содержимого динамических ОЗУ?
18. Охарактеризуйте основные сферы применения статических и динамических ОЗУ.
19. Какое влияние на асинхронный режим работы памяти оказывает синхронный характер работы контроллера памяти?
21. Какой вид ПЗУ обладает наиболее высокой скоростью перепрограммирования?
22. Какими методами обеспечивается энергонезависимость ОЗУ?
23. В чем состоит различие между режимами стандартной и запаздывающей записи в статических ОЗУ?
24. В чем проявляется специфика ОЗУ, предназначенных для видеосистем??

ПРАКТИЧЕСКАЯ РАБОТА № 4.

Изучение системы Windows Mobile. Изучение Windows Azure

Обзор Windows Mobile

Windows Mobile – одна из широко используемых в мире операционных систем для мобильных устройств, разработанная фирмой Microsoft. В данной ОС Microsoft удалось воспроизвести "в миниатюре" большинство функций *Windows* для настольных компьютеров. *Windows Mobile* работает на мобильных устройствах следующих классов: *Pocket PC*, смартфоны, коммуникаторы. Ее графический *интерфейс* напоминает настольные версии *Windows*. Многие традиционные для *Windows* приложения воспроизведены и в *Windows Mobile*, например, *Word Mobile*, *Excel Mobile* и т.д.

Наиболее новая версия *Windows Mobile* (*по* состоянию на осень 2010 г.) – 6.5.

Для лабораторной работы будем использовать версию 5.0, имеющуюся в нашем распоряжении.

Следует иметь в виду, что на мобильных устройствах для выбора информации на экране и для ее ввода используется прилагаемый к нему стайлус (*stylus*) – пластиковое "перо", которым необходимо легко касаться экрана. Касание стайлусом экрана в англоязычных руководствах и подсказках именуется термином "*tap*". В некоторых моделях возможен также выбор действия путем касания экрана пальцем.

Запуск системы

Включите *мобильное устройство* с операционной системой *Windows Mobile*.

Выдается стартовый *баннер*, затем (через 1-2 минуты) система предлагает Вам ввести *PIN*-код устройства. Введите его.

Начало работы с системой

После загрузки ОС на экране появляется стартовая страница ("Today" – "Сегодня") системы *Windows Mobile* (рис. 37.1):

Рис. 37.1. Стартовая страница (Today) Windows Mobile.

Проанализируйте информацию, выдаваемую на экране: текущая дата, *информация* о владельце, о полученных сообщениях, назначенных заданиях, неотвеченных звонках.

В левом верхнем углу (в отличие от настольной *Windows*) – кнопка Start (Пуск), при нажатии на которую выдается стартовое *меню*: *Интернет*, *Интернет*, Календарь, Сообщения, Телефон, Контакты, Калькулятор, Камера, Программы и др.

Запуск программ

Для визуализации списка программ нажмите Start / Programs (Пуск / Программы). На экран выводится *список* программ (рис. 37.2) :

Рис. 37.2. Список программ Windows Mobile

Здесь ActiveSync – *программа* для синхронизации *мобильного устройства* с персональным компьютером, *Download Agent – программа* для загрузки приложений в *мобильное устройство*, File Explorer – *программа* для *просмотра файлов* (аналог *Windows* Explorer для ПК), *Excel Mobile, PowerPoint Mobile* и др. – аналоги *компонент* Microsoft Office для ПК.

Вы как бы попадаете в знакомый мир приложений настольной версии *Windows* со знакомыми обозначениями приложений и файлов, что является одной из наиболее привлекательных особенностей *Windows Mobile*.

Поэкспериментируйте с вызовами *компонент* Microsoft Office *Mobile*.

Работа с файлами

Работа с файлами и папками (folders) – хранилищами ссылок на файлы и другие папки – осуществляется с помощью программы File Explorer (Проводник). На рис. 37.3 показано окно программы File Explorer:

Рис. 37.3. Программа File Explorer (Проводник).

Открытие файла или папки в директории осуществляется касанием его иконки стайлусом. При этом для файла выполняется действие его открытия, зависящее от его типа, - для текстовых файлов – вызов соответствующего редактора, для файлов *.pdf* – вызов аналога программы Adobe Acrobat, и т.д.

Поэкспериментируйте на своем компьютере с навигацией *по* файлам и папкам и открытием файлов с документами.

По умолчанию откроется *папка* **Мои Документы**.

Типичные действия – переход в родительской директории, передача файла и т.д. – выполняются с помощью *меню*, находящегося в нижней строке экрана.

Среди пунктов *меню* найдите особенно важные дляя *мобильного устройства* действия – передачу файла на *компьютер* через ИК-порт или *Bluetooth* на ПК или на другое *мобильное устройство*. Перешлите какой-либо *файл* на ПК через *Bluetooth*. Если потребуется, включите *Bluetooth* на Вашем устройстве (соответствующие пункты *меню* найдите самостоятельно).

Выход в Интернет

Выход в *Интернет* осуществляется, как и в настольной *Windows*, с помощью мобильного аналога программы *Internet* Explorer (рис. 37.4):

Рис. 37.4. Internet Explorer в ОС Windows Mobile

Для вызова *Internet* Explorer выберите Start / *Internet* Explorer.

Если подключение к Интернету не установлено, выберите Пуск / Настройки / Подключения / Подключения, где с помощью предложенного *меню* последовательно введите настройки для подключения к Интернету, *информация* о которых предоставляется Вашим провайдером мобильной связи. Система выдает подробные пояснения *по* поводу этих настроек.

В системе имеется также аналог Microsoft *Outlook* для приема и отправки электронной почты. Найдите его самостоятельно и поэкспериментируйте с ним.

Работа с рисунками и видео

Программа **Рисунки и Видео (Pictures and Videos)** предоставляет удобный *интерфейс* для работы с цифроввыми фотографиями и видеоклипами.

Для получения фотографий и снятия видео выберите **Пуск / Камера**.

Для запуска программы **Рисунки и Видео** выберите **Пуск / Программы / Рисунки и видео**.

В результате на экране в виде небольших картинок будет отображено содержимое Вашей папки **Мои рисунки** (рис. 37.5) :

Рис. 37.5. Программа Рисунки и Видео (Pictures and Videos)

Меню для управления программой – в нижней строке экрана. Выберите **Меню / Показ слайдов** и просмотрите содержимое директории **Мои Рисунки** как слайд-шоу. Чтобы его остановить, коснитесь экрана и нажмите "крестик" для выхода из окна просмотра.

В системе имеется также ряд традиционных программ – Календарь, Назначение заданий, а также аналоги *Word, Excel, PowerPoint*. Они в данной лабораторной работе подробно не рассматриваются. Поэкспериментируйте с ними самостоятельно.

Управление программами и памятью

Windows Mobile позволяет управлять памятью и вызванными в системе программами.

Чтобы использовать данную функцию системы, выберите **Пуск / Настройка / Система / Память**. Вы увидите три вкладки: **Оперативная, Карта памяти, Запущенные программы**.

Выбрав первую вкладку, узнайте общий объем оперативной памяти Вашего устройства.

Выбрав вторую, получите информацию о карте памяти, используемой в Вашем устройстве.

Выбрав третью, получите *список* запущенных программ (рис. 37.6) :

Рис. 37.6. Управление программами в Windows Mobile

Выбрав любую программу, Вы можете ее остановить или активировать. Возможна также остановка всех программ.

Управление контактами

Для *мобильного устройства* очень важно иметь возможность управлять списком контактов. ОС *Windows Mobile* такую возможность предоставляет.

Для просмотра и изменения списка контактов выберите **Пуск / Контакты**.
При вводе нового контакта на экран выдается форма, изображенная на рис. 37.7 :

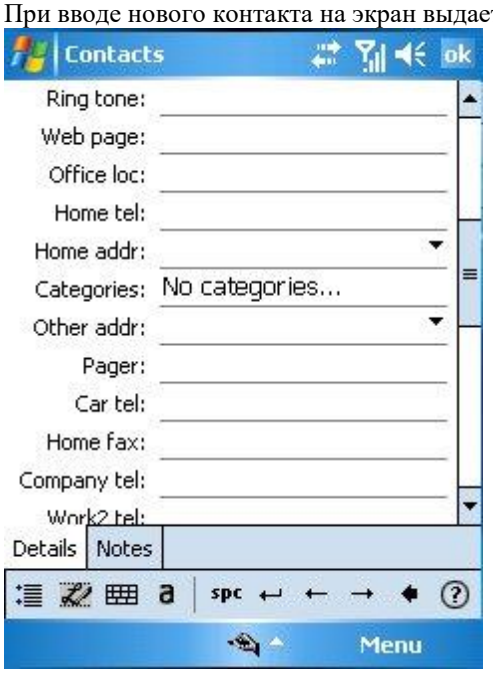


Рис. 37.7. Ввод нового контакта в список контактов

При отправке электронной почты, SMS-сообщения или при звонке *список* контактов может быть использован для выбора абонента-получателя.

Имеется возможность синхронизации списка контактов со списком контактов программы Microsoft *Outlook*, установоенной на Вашем настольном ПК.

Телефонные звонки

Для того, чтобы сделать исходящий звонок, выберите **Пуск / Телефон**. На экран выводится клавиатура мобильного телефона и *информация* об операторе мобильной связи (рис. 37.8):

Рис. 37.8. Форма для исходящих звонков в Windows Mobile

Для ответа на входящий звонок достаточно нажать любую клавишу.

Поэкспериментируйте с исходящими и входящими звонками.

Выход из системы

Для выхода из системы и выключения *мобильного устройства* нажмите клавишу выключения устройства. Система потребует от Вас подтверждения Ваших действий и *по* его получении выполняет выгрузку ОС и выключает устройство.

В данной лабораторной работе Вы познакомились лишь с некоторыми базовыми возможностями ОС *Windows Mobile*. Более подробно с ними, а также с методами программирования для *Windows Mobile* можно познакомиться в книге

Обзор Windows Azure

Целью лабораторной работы является практическое освоение Windows Azure– новейшей платформы Microsoft для облачных вычислений (cloud computing). Необходимый теоретический материал по основам облачных вычислений и основам архитектуры платформы Windows Azure представлен в "ОС для облачных вычислений (cloud computing). Windows Azure" данного курса. Работа в Windows Azure требует наличия академического доступа к Windows Azure, либо платной регистрации в ней. Альтернативой является использование бесплатной облачной системы Windows Live [20] , основанной на Windows Azure, что и учтено в данной лабораторной работе. При подготовке лабораторной работы использованы материалы [17-20].

Содержание

Аппаратура и программные инструменты, необходимые для лабораторной работы

Настольный или портативный компьютер с одной из новых версий операционной системы Microsoft Windows, в которой инсталлирован браузер Internet Explorer 8 или Internet Explorer 9.

Продолжительность лабораторной работы

2 академических часа

Обзор Windows Azure

Windows Azure – платформа для облачных вычислений фирмы Microsoft.

Облачные вычисления (cloud computing) - новая парадигма вычислений, основанная на идее использования через Web-браузер с любого компьютера набора специализированных сервисов, развернутых и доступных для использования через Web на компютерах мощного центра обработки данных.

Пользователь должен лишь иметь браузер и быть зарегистрированным на сайте для облачных вычислений.

Все остальное обеспечивается облачными сервисами, которые предоставляют программы, память для хранения данных пользователя, доступ к СУБД и другие вычислительные услуги.

Преимуществом такого подхода являются минимальные требования к компьютеру пользователя (для использования достаточно нетбука). Недостаток – полная зависимость пользователя от "облака", которое является единственным способом доступа не только к запускаемым программ, но и к собственным данным пользователя.

Архитектура Windows Azure описана в "ОС для облачных вычислений (cloud computing). Windows Azure" данного курса.

Вход на сайт платформы Windows Azure

Обратитесь через браузер по Web-ссылке [18] к сайту платформы Windows Azure.

Структура начальной страницы показана на рис. 38.1 .

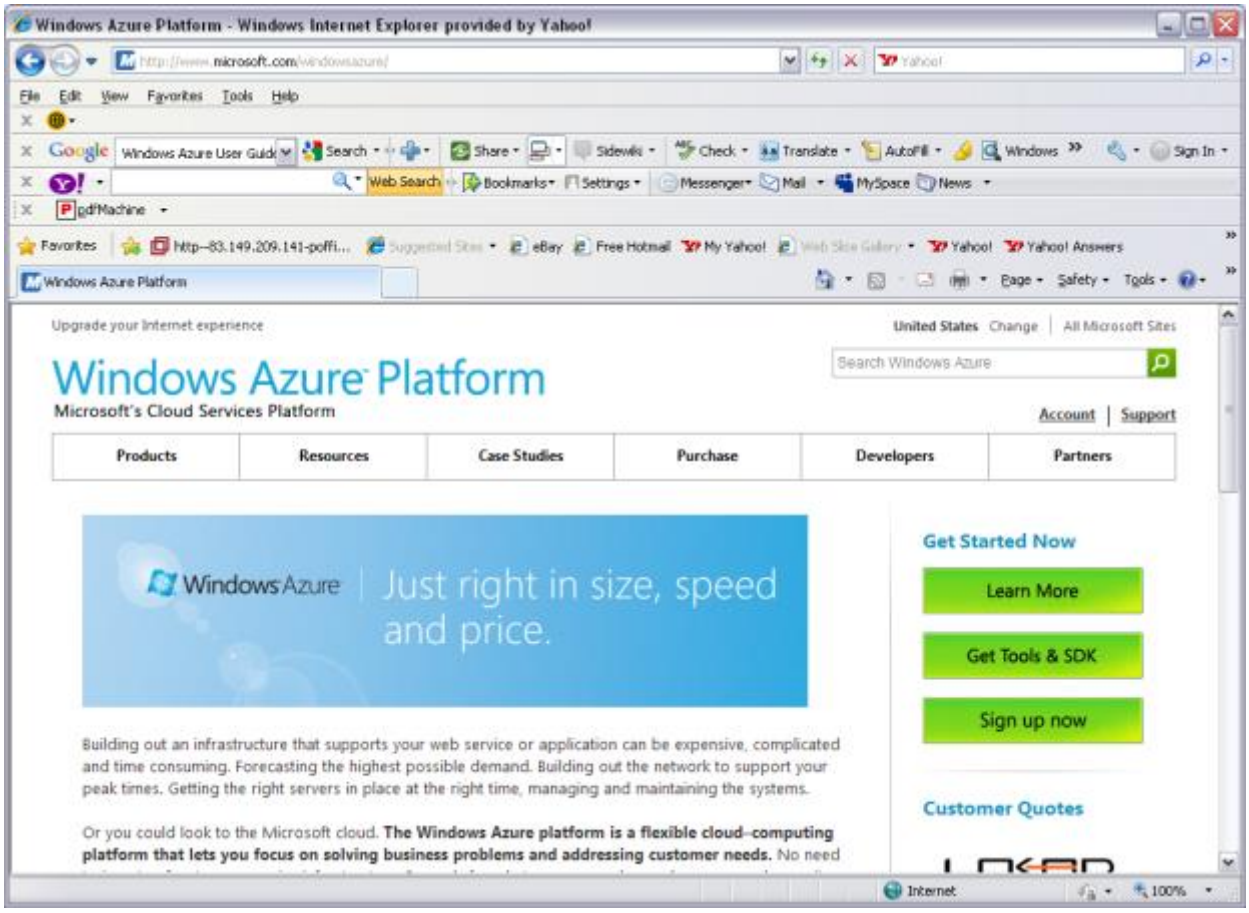


Рис. 38.1. Начальная страница веб-сайта платформы Windows Azure

На сайте доступна информация о продуктах, ресурсах, примерах использования, покупке (аренде) у фирмы Microsoft доступа к платформе Azure, а также ссылки для входа в Azure, регистрации в ней, скачивания инструментов и обучающих материалов по Azure.

Обзор платформы Azure на ее сайте

Нажмите Products (слева вверху). Визуализируется web-страница с кратким введением в платформу Azure (рис. 38.2):

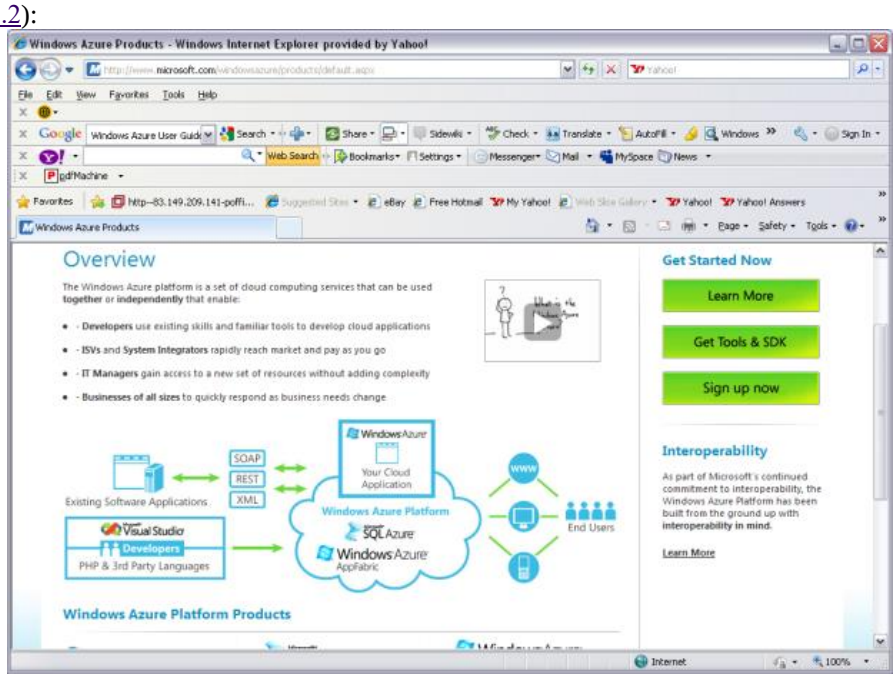


Рис. 38.2. Введение в платформу Azure на ее веб-сайте

Схема на рис. 38.2 поясняет особенности Windows Azure. Пользователи могут обращаться к ней через браузеры с настольных, портативных компьютеров и мобильных устройств. Приложения, работающие на платформе Azure, разрабатываются в среде Visual Studio. Их работа основана на платформе .NET, обеспечивающей надежное и безопасное исполнение кода. В частности, основой реализации Azure является компонента Windows Communication Foundation (WCF) и предоставляемые ею web-сервисы. Технологиями (стандартами), используемыми при реализации

Azure, являются XML (стандарт представления данных), SOAP (стандарт передачи данных через сеть с помощью "конвертов" в формате XML; REST – один из стандартов для управления web-сервисами.

Использование, архитектура и перспективы Windows Azure

Схема использования Windows Azure изображена на рис. 38.3 .

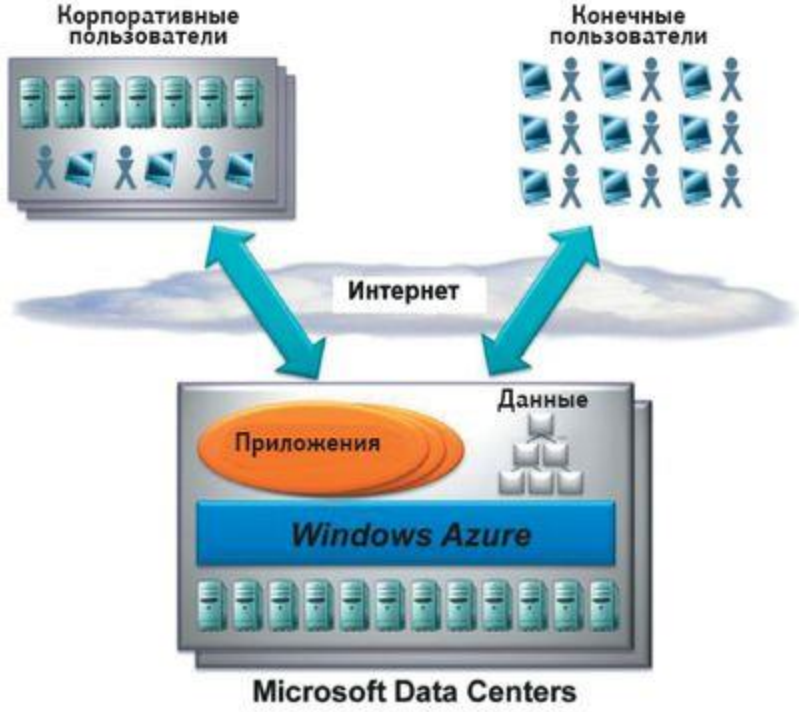


Рис. 38.3. Схема использования Windows Azure

Осенью 2008 г. Microsoft объявила о создании новой облачной операционной системы Windows Azure, предназначенной для разработки облачных приложений. Тем самым, Microsoft начала третью эру операционных систем [19] в надежде повторить успех DOS в 1980 х гг. и Windows в 1990 х. В 2010 году Windows Azure объявлена коммерческой системой. Как и традиционная ОС, Windows Azure позволяет запускать приложения и хранить данные. Но происходит это не на компьютере пользователя, а в вычислительных облаках.

Операционная система Windows Azure является частью Windows Azure Platform — группы облачных технологий для разработки ПО, которая включает следующие элементы:

- Windows Azure - обеспечивает Windows-среду для работы приложения и хранения данных в дата-центрах Microsoft.
- SQL Azure - обеспечивает работу с реляционными базами данных на основе SQL-сервера. Данные могут храниться как в облачной среде, так и на компьютерах предприятия, тем не менее, взаимодействуя с приложениями Windows Azure.
- Windows Azure Platform AppFabric - объединяет приложения, работающие как в облачной, так и в традиционной среде, обеспечивая защищенную передачу данных. Несмотря на сходство названий, понятия fabric и AppFabric —не одно и то же. Первое относится к объединению физических машин внутри облачной ОС, второе — к соединению приложений, работающих в разных средах.

Непосредственно операционная система Windows Azure также состоит из нескольких взаимосвязанных частей: Compute Service, Storage Service и Fabric.

Компонента Compute Service отвечает за вычисления. Основная цель облачной платформы состоит в том, чтобы обеспечить поддержку приложения, запускающего огромное число пользователей в одно и то же время. Windows Azure поддерживает несколько копий одного и того же кода на разных физических серверах. В свою очередь, приложение может работать сразу в нескольких версиях на нескольких виртуальных машинах, каждая из которых обеспечивается гипервизором на основе Hyper-V, модифицированным для использования в облачных вычислениях.

Существуют два типа рабочих версий облачного приложения: веб-роль (Web role) и рабочая роль (Worker role). Первая умеет обрабатывать HTTP- или HTTPS-запросы, и на ее виртуальной машине (ВМ) запущен сервер Internet Information Services (IIS). Программист имеет возможность создать версию веб-роли с помощью ASP.NET либо Windows Communication Foundation (WCF), а также воспользоваться любой другой технологией .NET, работающей с IIS. Приложение может быть создано на любом языке программирования.

Напротив, рабочая роль не предполагает запуска IIS. Она выполняет задачи в фоновом режиме. Например, веб-роль может быть применена для получения запроса от пользователя. Но его обработка будет запущена позже с помощью версии рабочей роли.

Архитектура Windows Azure изображена на рис. 38.4 .

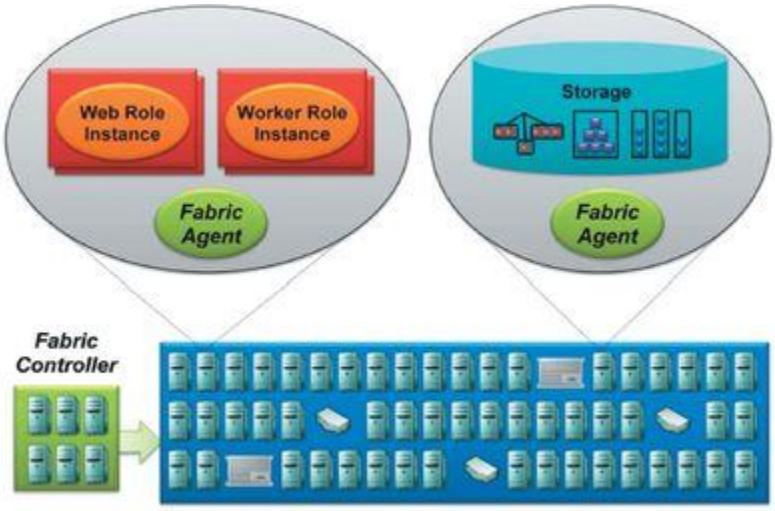


Рис. 38.4. Архитектура Windows Azure

Компонента Storage Service обеспечивает хранение данных. ОС Windows Azure поддерживает три способа работы с данными. Самый простой из них — BLOB (binary large object), содержащий бинарные данные с несложной иерархией. Этот тип организации информации предназначен для хранения изображений, аудио- и видеоинформации.

Если необходимо структурировать однотипные данные, то используются таблицы, в которых для каждой единицы информации отведена ячейка с определенным номером строки и номером столбца. Столь простая организация позволяет получать доступ к данным посредством методов ADO.NET. В таком виде облачная ОС распределяет хранение данных на несколько физических компьютеров, что более эффективно, чем при использовании реляционной базы данных.

Рассмотренные способы обеспечивают хранение данных и доступ к ним, а для их связи необходим третий способ – очередь (FIFO). Этот способ помогает разным версиям приложения обмениваться между собой сообщениями. Так связываются веб-роль и рабочая роль, поскольку синхронизация в облачной среде невозможна. Предположим, пользователь через веб-интерфейс вызывает задачу, требующую существенных вычислительных мощностей. Веб-роль записывает полученный запрос в очередь. Рабочая роль, обращаясь к этой очереди, принимает запрос и выполняет его. Результаты выполнения (ответ) передаются по тому же принципу, через очередь.

Независимо от метода организации данных, информация в Windows Azure Storage реплицируется 3 раза, что обеспечивает устойчивость системы: потеря данных в одной из копий не фатальна. Кроме того, существуют архивные копии, хранящиеся в другом дата-центре Microsoft. Это означает, что даже если весь дата-центр уничтожен, информация будет восстановлена из архивов другого центра.

Последняя составляющая ОС — Fabric — позволяет организовать набор компьютеров, на которых хранятся приложения и данные Windows Azure. Управление такой "компьютерной тканью" осуществляет программное обеспечение, называемое fabric controller.

Fabric осуществляет мониторинг всех работающих приложений, управляет взаимодействием с ОС на разных компьюьтерах и выбирает физический сервер для запуска приложения, тем самым оптимизируя использование оборудования.

Управление приложениями выполняется с помощью конфигурационных файлов, содержащих XML-описание всего, что необходимо приложению, например, нужного количества виртуальных машин с веб-ролями и рабочими ролями. Fabric controller создает эти виртуальные машины и отслеживает состояние каждой из них, чтобы при необходимости заменить вышедшую из строя или запустить ее на другом физическом сервере.

Компоненты Windows Azure позволяют строить приложения разных типов. Так, для создания масштабируемого интернет-приложения программисту достаточно употребить необходимое количество веб-ролей, сохраняя данные в таблицах. А для приложения с параллельными вычислениями потребуются веб-роль, очередь для сохранения запросов, необходимое количество рабочих ролей и таблицы (или BLOB) для хранения данных. В свою очередь, SQL Azure и AppFabric дают возможность соединить решения Windows Azure с программами и базами данных, функционирующими в рамках локальной сети или с облачными системами других провайдеров.

Приложения, созданные на основе Windows Azure, предоставляются как сервисы физическим лицам, корпоративным пользователям или и тем, и другим одновременно. Ориентировочные цены на некоторые облачные услуги Microsoft приведены в статье [19] .

Рассмотрим примеры облачных приложений, созданных с помощью Windows Azure и рассчитанных на разные типы пользователей.

Решение для корпоративных пользователей. С помощью Windows Azure независимый разработчик программного обеспечения может создавать приложения для бизнес-пользователей, применяя принципы программного обеспечения как сервиса (SaaS — Software as a Service).

Примером может послужить решение, разработанное американской компанией Alinean, Inc. Ее сфера деятельности — предоставление по запросу аналитических средств в области анализа продаж и маркетинга. Системы Alinean позволяют оценить потребности и возможности бизнеса в будущем, предложить решение для наращивания мощностей и подсчитать, когда начнут окупаться инвестиции. Пользователями Alinean являются корпоративные

клиенты, находящиеся в разных уголках земного шара. Среди них IBM, HP, Microsoft, Intel, AT&T, VMware, Oracle, Siemens, Symantec и др. В дата-центре Alinean, находящемся в Орландо (Флорида, США), сервис по запросу предоставляли 20 серверов, работающих 24 часа в сутки семь дней в неделю. Объем бизнеса рос, и мощностей стало не хватать, да и содержание внутреннего ЦОД становилось все дороже.

Поэтому было принято решение перенести разработанное ранее программное обеспечение под крышу Windows Azure. В результате потребовалось 28 виртуальных серверов с Azure и 20 SQL Azure (по 10 Гбайт каждый). Благодаря этому, Alinean удалось добиться сокращения затрат по обслуживанию на 60% по сравнению с предыдущей, традиционной моделью. Кроме того, руководство оценивает в 160% отношение среднего увеличения прибыли к объему инвестиций (ROI — Return On Investment) в Windows Azure по сравнению с вложениями в прежнюю конфигурацию (100%)

Решение для физических лиц. Благодаря масштабируемости Windows Azure позволяет вести учет огромного количества пользователей. Создавая облачное решение, компания-разработчик может рассчитывать не только на корпорации, но и на физических лиц.

Такое приложение было сделано новозеландской компанией TicketDirect International, которая, работая в онлайновом режиме, осуществляет 45?% всех продаж билетов на культурные и спортивные мероприятия Новой Зеландии. Предыдущая, традиционная, система продажи билетов, функционировавшая на базе Microsoft SQL Server 7 и SQL Server 2000, была написана на Visual Basic 6. Приложение обслуживало несколько сотен продаж в течение часа. Но в дни распродаж, когда объявлялась скидка на посещение популярного мероприятия, в систему пытались войти тысячи людей. Не удивительно, что компьютерный центр продавца билетов не выдерживал такого наплыва пользователей.

Windows Azure предоставила TicketDirect масштабируемую инфраструктуру как сервис с возможностью оплаты по факту. В результате в момент распродаж приложение начинает использовать дополнительные мощности. Теперь компании TicketDirect не потребуется закупать оборудование только для того, чтобы покрыть временные всплески активности. Ограничений практически не существует. В облаках компания способна обслужить несколько популярных мероприятий, начинающих свои распродажи в одну и ту же минуту. Windows Azure предоставит любыемощности, необходимые для бизнеса, исследований, обучения.

Внутреннее решение. В среде Windows Azure могут быть созданы внутренние приложения, пользователями которых являются работники данного предприятия. В этом случае масштабируемость не столь важна.

В качестве примера приведем саму компанию Microsof - ее отдел информационных технологий, где нашла свое применение Windows Azure. В рамках ежегодной благотворительной кампании ИТ-отдел проводит онлайн-аукцион в пользу благотворительной организации United Way. Прежде оборудование и ПО для него поддерживались круглый год, в то время как мероприятие проводилось в течение одного месяца всего лишь раз в году. Кроме того, в самом конце аукциона обычно возникала еще одна проблема, с которой сталкивались технические работники. Каждый раз в это время наблюдался всплеск активности, и система оказывалась перегруженной.

Отдел ИТ принял решение мигрировать в вычислительные облака. Были использованы Windows Azure и Microsoft SQL Azure для хранения данных. Теперь в последние дни аукциона ИТ-команда программирует систему на использование большего количества ресурсов, чтобы обслужить увеличивающийся поток запросов. Когда аукцион заканчивается, мощности сокращаются соответственно нагрузке. Облачная модель готова обслужить столько пользователей, сколько необходимо. Внутри огромной компании, которой является Microsoft, система теперь позволяет собрать больше средств, идущих на благотворительность.

Приведенные примеры иллюстрируют возможности создания систем по запросу. Но для того чтобы поработать в среде Windows Azure, не обязательно программировать свое собственное приложение. Сейчас каждый из нас сумеет протестировать облачную ОС Microsoft в действии. На базе Windows Azure в рамках "живой", работающей системы Windows Live [20] доступны офисные приложения по запросу. Windows Live позволяет создавать документы в форматах Word, Excel и PowerPoint и хранить их на виртуальном диске, в облаках. Любопытно, что система дает возможность открыть онлайн-документ на ПК с помощью традиционного ПО Microsoft.

В будущем Windows Azure выйдет за пределы дата-центров ее разработчика и будет устанавливаться в стенах других корпораций. Microsoft объявила о предстоящем взаимодействии с такими компаниями, как Dell, HP и eBay. Последняя планирует использовать облачное решение на основе Windows Azure, благодаря чему абоненты смогут участвовать в привычном аукционе eBay, используя iPad.

По словам Стива Балмера, наступил один из важнейших моментов компьютерной эры. "Все понимают, что облачные вычисления чрезвычайно выгодны, и это открывает большие возможности…"

Ознакомление с Windows Live

Windows Live – бесплатная облачная платформа на основе Windows Azure, предоставляющая облачные сервисы для пользователей.

Для использования Windows Live зарегистрируйтесь с помощью службы Microsoft Passport.

Войдите на сайт Windows Live (рис. 38.5):

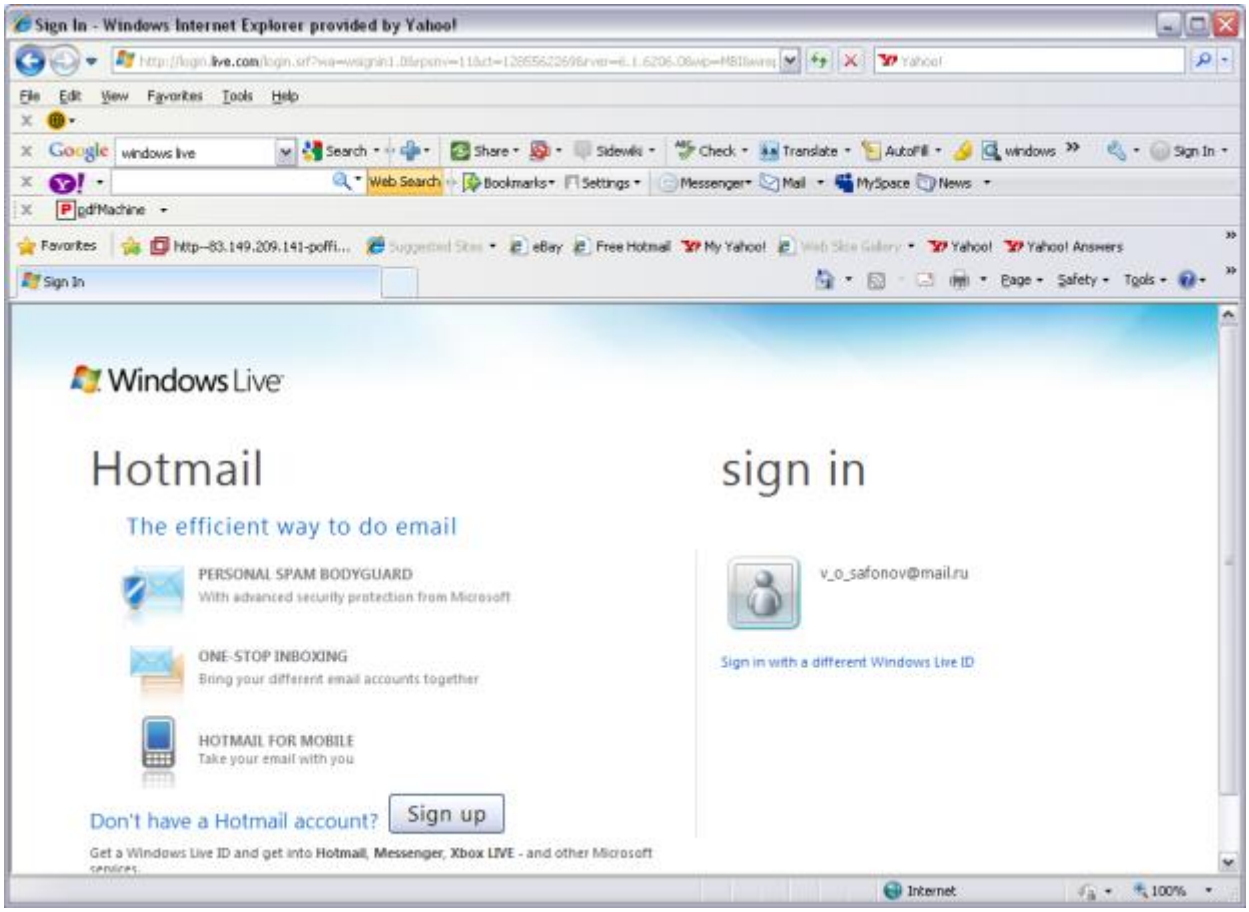


Рис. 38.5. Сайт Windows Live

Затем войдите в систему под своим именем. Визуализируется страница сервисов, представленная на рис. 38.6:

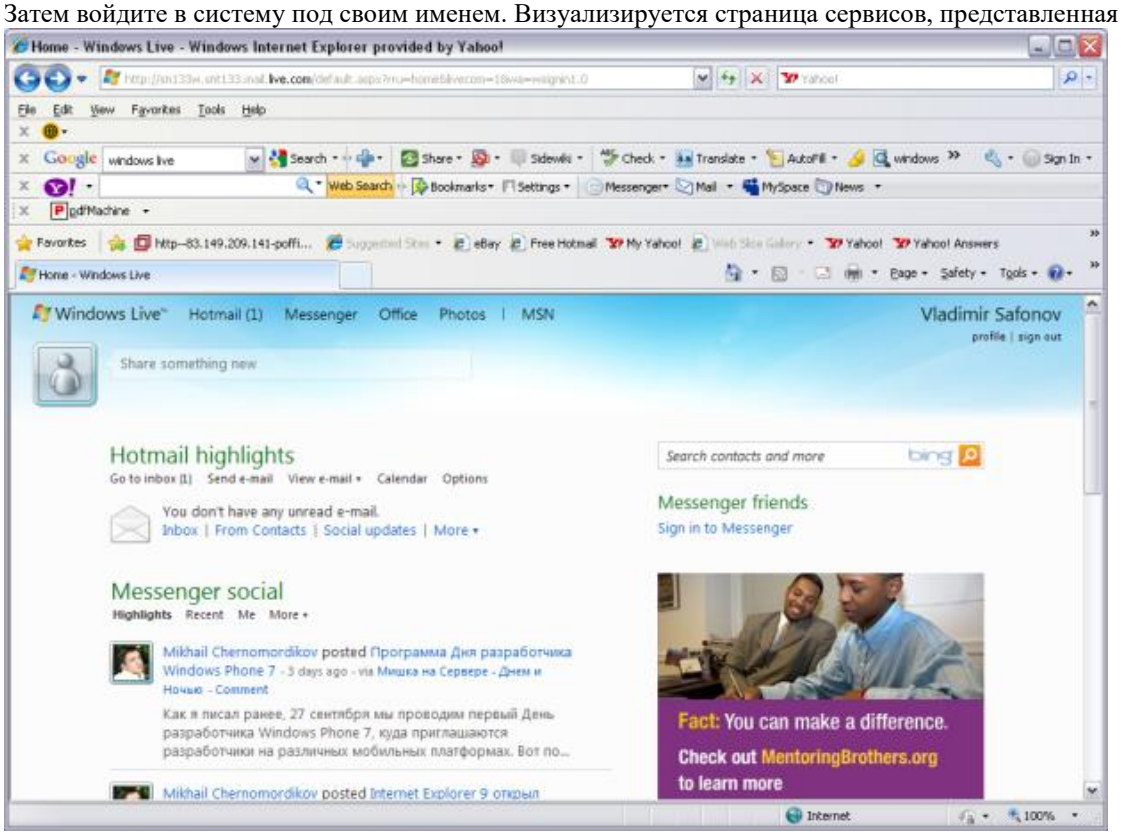


Рис. 38.6. Стартовая страница сервисов Windows Live

С помощью этой страницы воспользуйтесь облачными сервисами. Нажмите **Office** для создания офисных документов с пмощью облачных сервисов (рис. 38.7):

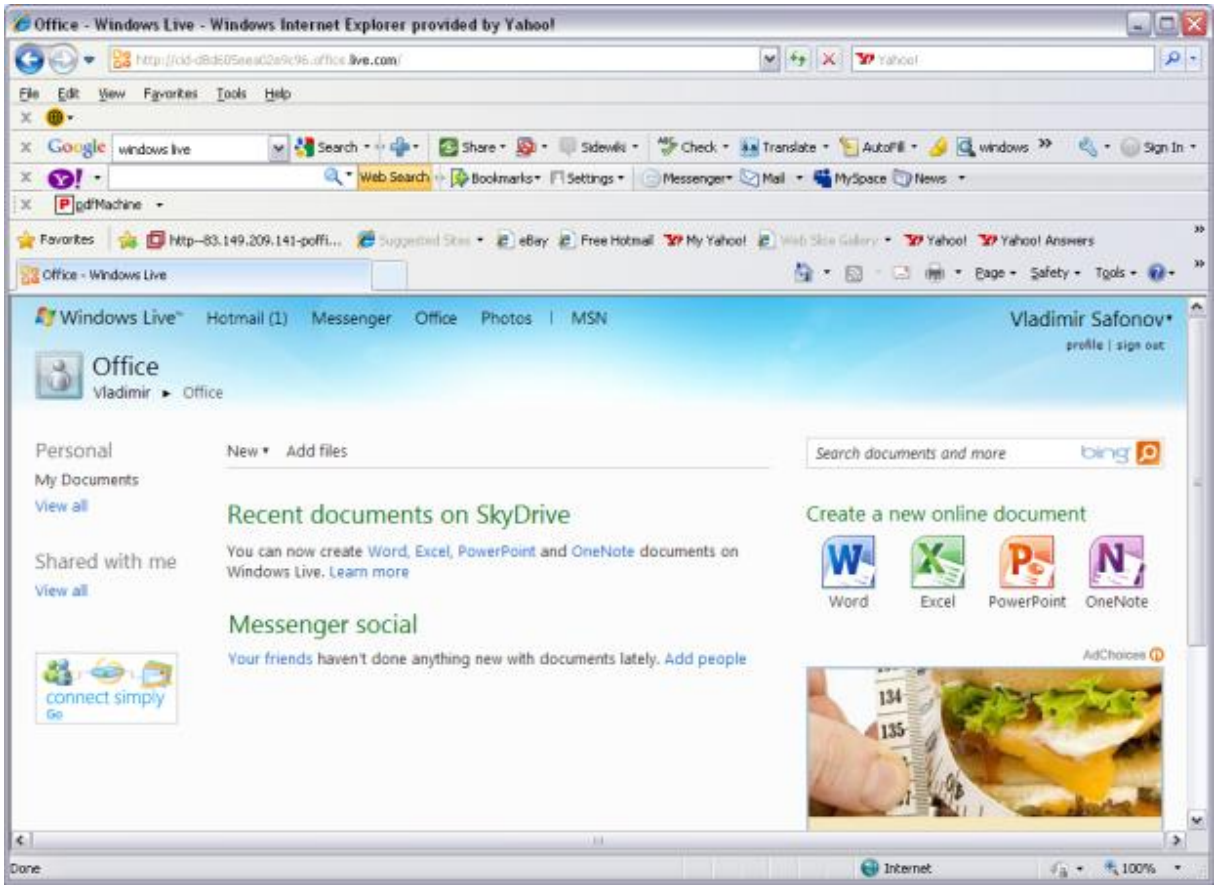


Рис. 38.7. Создание офисных документов в Windows Live

Заметьте, что для этого Вам нет необходимости инсталлировать MS Office: благодаря принципам облачных вычислений и Windows Azure, Вы можете создать офисный документ.

Выберите Create a new office document / Word. Система создаст Вам пустой документ Document1.docx. Затем Вы попадаете на страницу, с помощью которой Вы создаете документ (рис. 38.8):

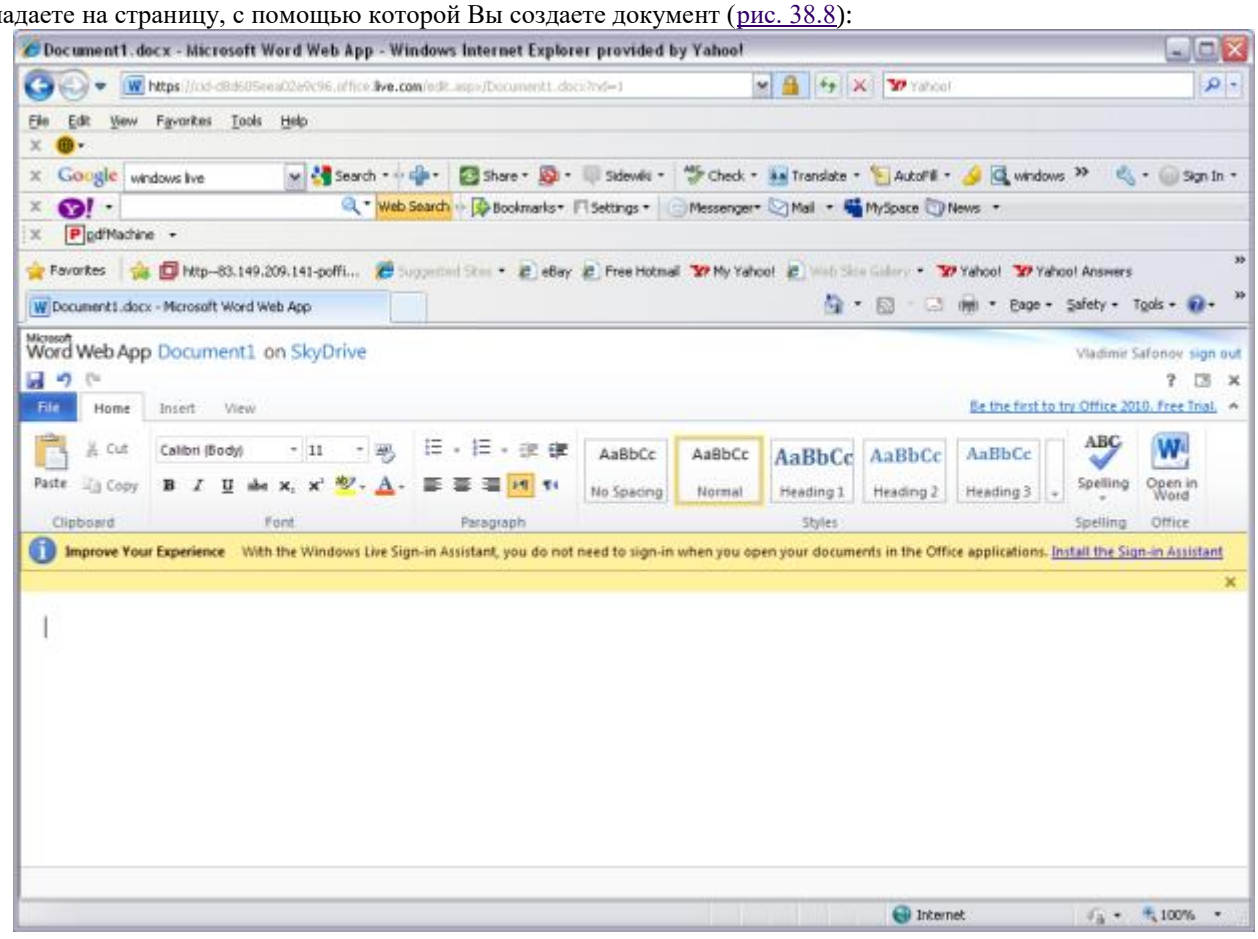


Рис. 38.8. Работа с документом MS Word через “облако” в Windows Live

Наберите текст документа**: Это мой первый облачный сервис!** (рис. 38.9):

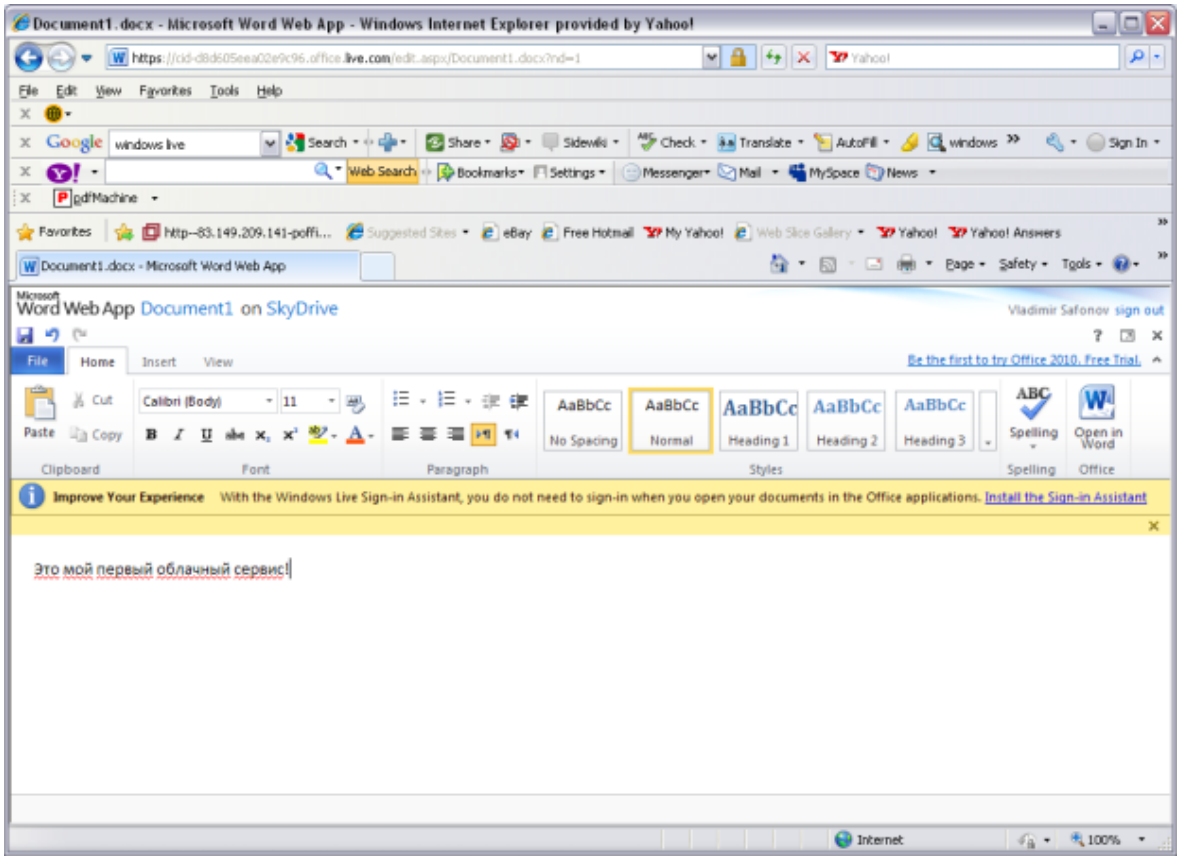


Рис. 38.9. Ваш первый “облачный” офисный документ

Затем выберите File / Save (как Вы сделали бы в MS Word – интерфейс почти идентичен). Выйдите из режима просмотра документа, нажав "X". Визуализуется страница, позволяющая Вам продолжить работу с офисными документами (рис. 38.10):

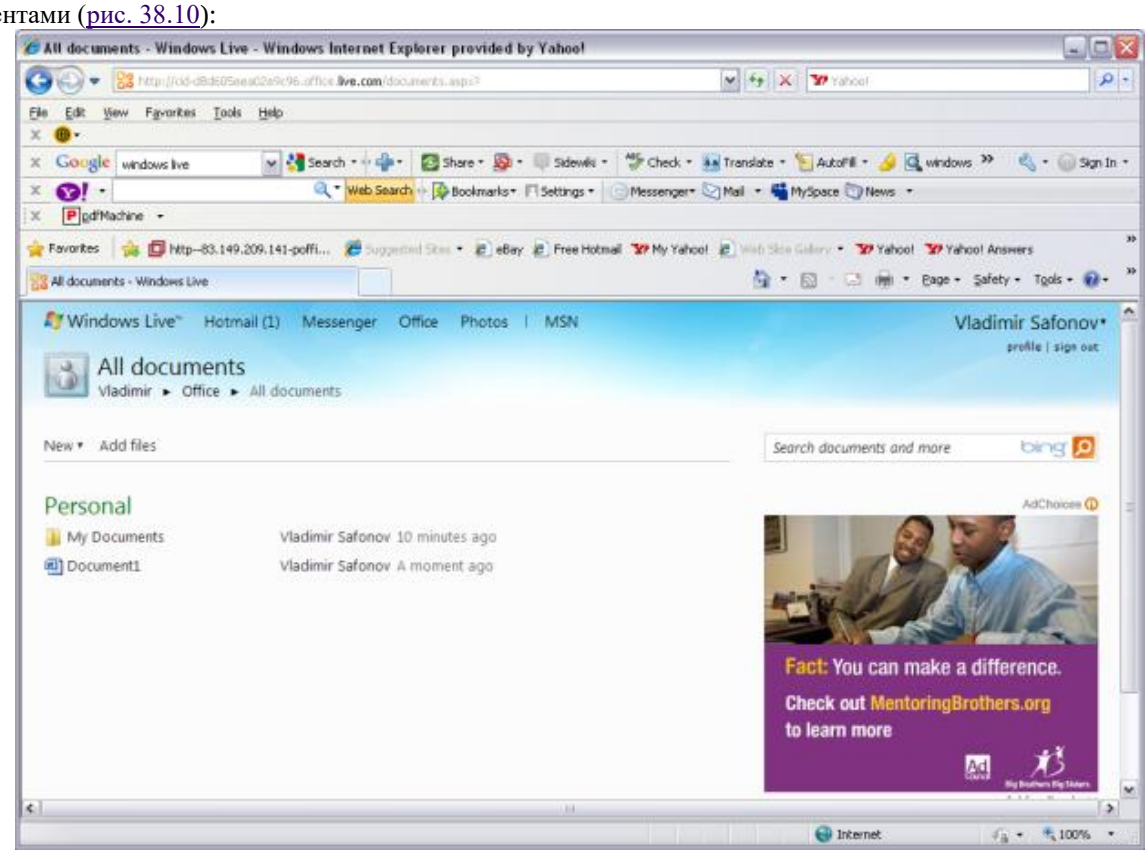


Рис. 38.10. Страница Windows Live для работы с Вашими документами

Затем поэкспериментируйте с повторным входом в Document1, убедитесь, что он сохранился на машинах дата-центра Microsoft (в облаке).

Для **выхода** из системы Windows Live нажмите Sign out.

Надеемся, что даже на таком простом примере Вы почувствовали преимущества облачных вычислений. Желаем Вам дальнейшей успешной работы в облаках Windows Azure и Windows Live!

Более подробная информация – в источниках [17-20], и на сайте MSDN.

Контрольные вопросы ПЗ4 (ПК-1):

1. Что такое диспетчеризация процессора?(ПК-1).
2. В чем основная цель диспетчеризации процессора?(ПК-1).
3. Что такое цикл CPU – I/O?(ПК-1).
4. Как зависит частота периодов активности процессора от их длительности?(ПК-1).
5. Что такое планировщик?(ПК-1).
6. Какие разновидности стратегий, с точки зрения прерывания или избежание прерывания процессов, использует планировщик?(ПК-1).
7. Что такое стратегия без прерывания процессов?(ПК-1).
8. Что такое стратегия с прерыванием процессов?(ПК-1).
9. Что такое диспетчер?(ПК-1,).
10. Что такое латентность диспетчера и каким образом следует оптимизировать данный показатель?(ПК-1,).
11. Каковы основные критерии диспетчеризации?(ПК-1,).
12. Что такое использование (утилизация) процессора и как следует оптимизировать данный показатель?(ПК-1,).
13. Что такое пропускная способность системы и как следует оптимизировать данный показатель?(ПК-1,).
14. Что такое время обработки и как следует оптимизировать данный показатель?(ПК-1,).
15. Что такое время ожидания и как следует оптимизировать данный показатель?(, ПК-1).
16. Что такое время ответа и как следует оптимизировать данный показатель?(, ПК-1).
17. Что такое диаграмма Ганта?(, ПК-1).
18. В чем суть стратегии FCFS и каковы ее недостатки?(, ПК-1).
19. В чем суть стратегии SJF (и SRTF) и оптимальность по какому критерию она обеспечивает?(, ПК-1).
20. Каким образом и по каким формулам вычисляется предсказание длины следующего периода активности процессора?(, ПК-1).
21. В чем суть диспетчеризации по приоритетам?(, ПК-1).
22. Что такое проблема голодания процессов и каково ее решение в ОС?(ПК-1).
23. В чем суть стратегии RR, оптимальность по какому критерию она обеспечивает и по какому критерию она хуже, чем SJF?(ПК-1).
24. Как зависит число контекстных переключений от величины кванта времени?(ПК-1).
25. Как зависит время оборота от величины кванта времени?(ПК-1).
26. Что такое многоуровневая аналитическая очередь и процессы каких классов обрабатываются с помощью многоуровневых очередей?(ПК-1).
27. Каковы особенности планирования загрузки многопроцессорных систем?(ПК-1).
28. Каковы особенности планирования в системах реального времени?(ПК-1).
29. В чем заключается задача управления памятью?(ПК-1).
30. Что такое входная очередь заданий?(ПК-1).
31. Что такое связывание адресов и на каких этапах обработки программы оно может выполняться?(ПК-1).
32. Какие этапы обработки проходит программа на пути от исходного кода к двоичному образу в памяти?(ПК-1).
33. Что такое компиляция?(ПК-1).
34. Что такое редактирование связей?(ПК-1).
35. Что такое загрузка?(ПК-1).
36. Что такое линковка?(ПК-1,).
37. Что такое объектный модуль?(ПК-1,).
38. Что такое таблица символов?(ПК-1,).
39. Что такое загрузочный модуль?(ПК-1,).
40. Что такое библиотека?(ПК-1,).
41. Что такое бинарный образ программы в памяти?(ПК-1,).
42. Что такое редактор связей?(ПК-1,).
43. Что такое загрузчик?(ПК-1,).
44. Что такое ассемблер?(, ПК-1).
45. Что такое логический адрес, какой компонентой системы он генерируется?(,ПК-1).
46. Что такое физический адрес, какой компонентой системы он генерируется?(,ПК-1).
47. Что такое устройство управления памятью?(, ПК-1).
48. Что такое регистр перемещения?(, ПК-1).
49. Что такое перемещаемый код?(ПК-1).
50. Что такое динамическая загрузка?(ПК-1).
51. Что такое динамическая линковка?(ПК-1).
52. Что такое статически линкуемая библиотека?(ПК-1).
53. Что такое динамически линкуемая библиотека?(ПК-1).
54. Что такое оверлейная структура программы?(ПК-1).
55. Что такое драйвер оверлея?(ПК-1).

ПРАКТИЧЕСКАЯ РАБОТА № 5.

Изучение Windows Research Kernel

Целью лабораторной работы является практическое освоение Windows Research Kernel (WRK) – исходных кодов исследовательского ядра Windows, предназначенных для более глубокого изучения архитектуры Windows и исследований в области операционных систем. Необходимый вводный и общий теоретический материал по архитектуре и особенностям WRK представлен в "Академическая программа Microsoft Shared Source Initiative. Открытое ядро Windows для изучения и исследований (Windows Research Kernel)" данного курса. Данная лабораторная работа является лишь начальным практическим ознакомлением с пакетом WRK, использование которого в обучении операционным системам само по себе может стать основой для семестрового или полугодового курса и (или) семинара по ОС. Методы использования WRK, а также основы внутренней архитектуры Windows описаны в книге [7]. Пакет WRK доступен для скачивания преподавателями, аспирантами и студентами, зарегистрированными с помощью Microsoft Passport, по ссылке [21] с академического сайта Microsoft. Размер дистрибутива для скачивания – 14 мегабайт.

Содержание

Аппаратура и программные инструменты, необходимые для лабораторной работы

Настольный или портативный компьютер с одной из версий операционной системы Microsoft Windows

Продолжительность лабораторной работы

2 академических часа

Обзор Windows Research Kernel

Пакет WRK включает исходный код ядраWindows XP x64 и Windows Server 2003 SP1 с окружением для сборки и тестирования экспериментальных версий ядра Windows для использования в целях изучения и преподавания.

WRK включает исходные коды для следующих компонент:

- Processes – Процессы
- Threads -Потоки
- LPC – Локальные вызовы процедур
- Virtual memory – Виртуальная память
- Scheduler - Планировщик
- Object manager – Менеджер объектов
- I/O manager – Менеджер ввода-выводв
- Synchronization - Синхронизация
- Worker threads – Рабочие потоки
- Kernel heap manager – Менеджер кучи ядра
- Прочая функциональность ядра (NTOS)

Пакет WRK полезен при разработке проектов, позволяющих студентам исследовать принципы операционной системы с использованием исходных кодов ядра. Он поддерживает построение экспериментов и проектов на основе модификации ядра Windows , обеспечивая высокоуровневые методы обучения и исследования и лучшее понимание архитектуры и реализации Windows.

WRK - Детали

Пакет Windows Research Kernel содержит исходные коды ядра Windows (NTOS).

NTOS реализует основные функции ОС для:

- Управления процессами
- Управления потоками

- Управления виртуальной памятью и кэш-памятью
- Управления вводом-выводом
- Управления реестром
- Функций исполнительной подсистемы ядра (executive), таких, как куча ядра и синхронизация
- Менеджера объектов
- Механизма локального вызова процедур
- Монитора безопасности
- Низкоуровневого управления процессором (планирование потоков, асинхронные и отложенные вызовы процедур, обработка прерываний, обработка исключений)

Компонента Hardware Abstraction Layer, файловые системы, сетевые стеки и драйверы устройств реализованы отдельно от NTOS и загружаются в режиме ядра как динамически линкуемые библиотеки. Исходные коды для этих динамических компонент не включены в WRK. Однако некоторые из них доступны в различных инструментальных наборах, опубликованных фирмой Microsoft, таких, как Installable File System Kit и Windows Driver Development Kit.

Пакет WRK включает значительную часть исходных кодов ядра NTOS из самых ноых версий Windowsподдерживающих архитектуру x64. Исходные коды ядра, не включенные в WRK, относятся, главным образом, к компонентам Plug and Play (динамическому подключению устройств), управлению электропитанием, верификатору устройств, интерфейсу отладки ядра и виртуальной машине DOS (выполняющей DOS-приложения).

Пакет WRK предназначается для преподавателей, работающих в области операционных систем, разрабатывающих курсы и учебники и желающих включить в них информацию о ядре Windows, базирующуюся на реальных исходных кодах.

Пакет WRK включает окружение для сборки и тестирования и бинарные коды для отсутствующих компонент, которые могут быть использованы для сборки полнофункционального ядра NTOS, инсталлируемого в системах Windows Server 2003 для x86/x64 и Windows XP x64.

Скачивание и установка WRK

Скачайте по ссылке [2] пакет Windows Research Kernel и распакуйте полученный архив.

Структура директорий WRK

Войдите в базовую директорию WindowsResearchKernel-WRK.

Вы увидите в Windows Explorer структуру директорий пакета WRK (рис. 39.1):

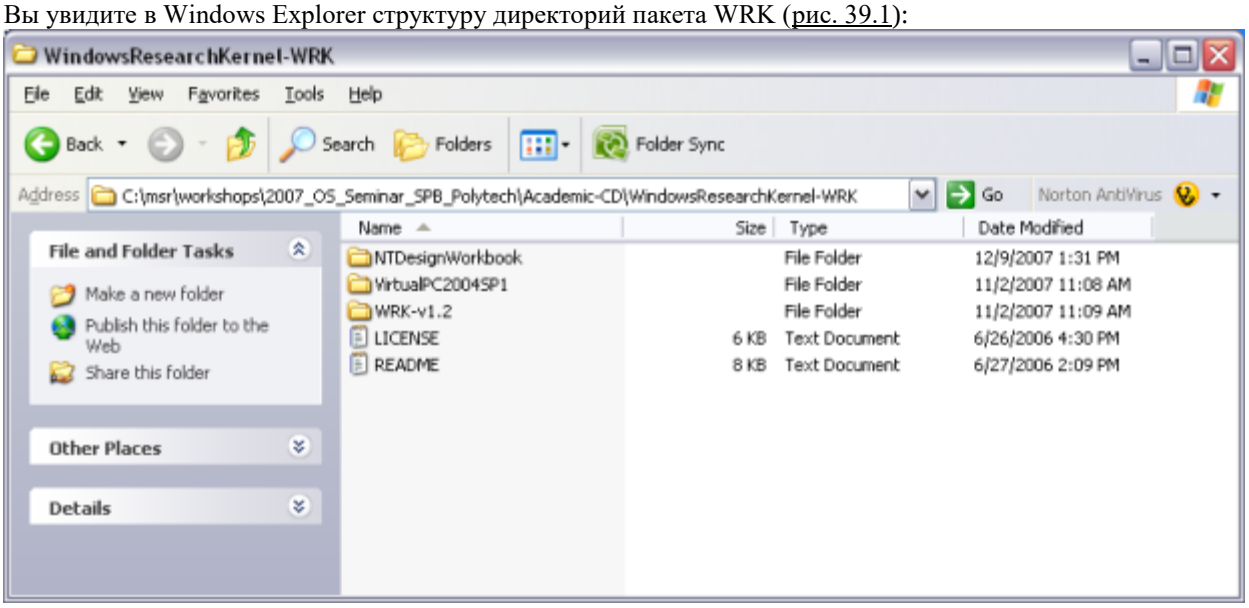


Рис. 39.1. Структура директорий пакета WRK

Файл README содержит краткое описание структуры пакета.

Файл LICENSE содержит подробное описание лицензии, на основе которой доступен WRK.

Кратко, суть лицензии в том, что WRK может быть использован только для обучения и исследований, но не джля коммерческих разработок.

Откройте файлы README и LICENSE и ознакомьтесь с их содержимым.

Весь материал для изучения в пакете WRK представлен на английском языке.

Ознакомьтесь с содержимым базовых директорий пакета WRK:

1. Директория NTDesignWorkbook содержит уникальный материал – фактически это подробные спецификации архитектуры всех компонент ядра Windows. Это – "святая святых" фирмы Microsoft, как и сами исходные коды ядра Windows. Приведение в доступный для изучения вид этих спецификаций для академического сообщества программистов потребовало, по признанию специалистов Microsoft, нескольких лет работы.
2. Директория VirtualPC2004S1 содержит дистрибутив инструмента Microsoft под названием Microsoft Virtual PC. Данный инструмент позволяет организовать на Вашем компьютере виртуальную машину, в которую Вы можете инсталлировать любую операционную систему, в том числе – экспериментальную ОС, являющуюся результатом Ваших экспериментов в WRK. В качестве отдельного практического занятия, можете инсталлировать

Microsoft Virtual PC и в полученную виртуальную машину инсталлируйте другую операционную систему, например, другую версию Windows или какую-либо версию Linux.

3. Директория WRK-v1.2 содержит собственно исходные коды ядра Windows. Их струкуру подробнее рассмотрим немного позже.

Ознакомление со структурой исходных кодов ядра Windows

Войдите в директорию WRK-v1.2.

Изучите структуру директорий ядра, изображенную на рис. 39.2 , краткое описание которой приведем ниже.

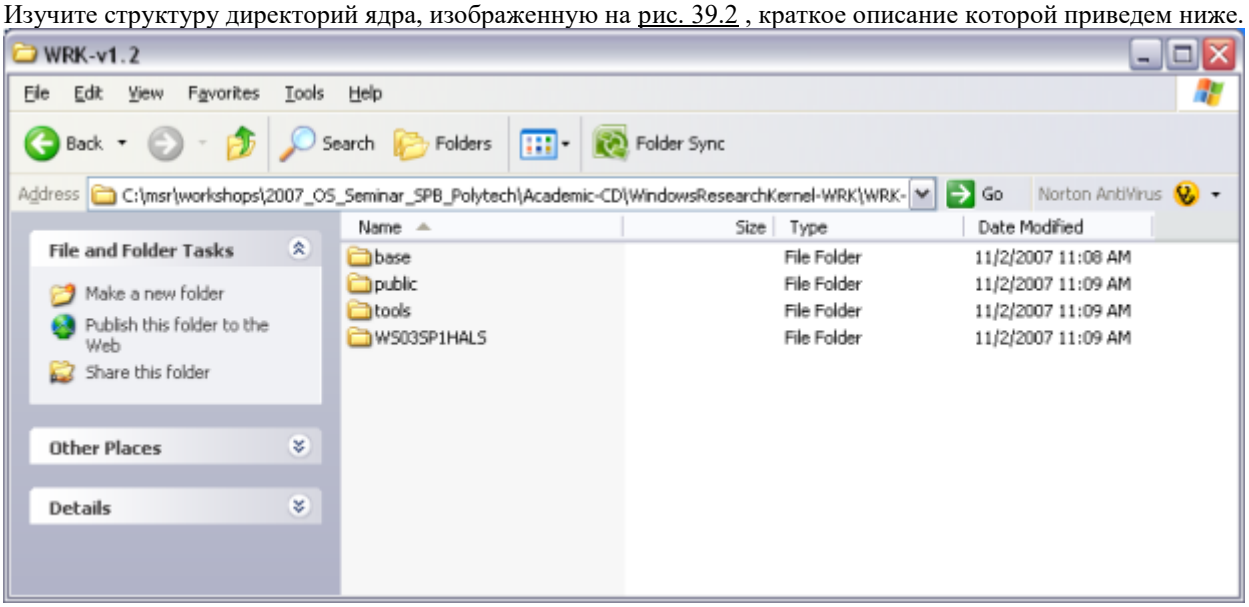


Рис. 39.2. Структура директорий исходного кода исследовательского ядра Windows

Директория **public** содержит include (заголовочные, .h) – файлы, используемые во всем исходном коде системы.

Например, заголовочный файл **public\ddk\inc\mountdev.h** содержит описание интерфейсов между точкой монтирования (mount point) и монтируемыми устройствами. Ознкомьтесь с содержимым файла. Вы убедитесь, что он (как и другие коды WRK) хорошо самодокументирован, однако для более глубокого понимания требуется подробное изучение на основе книги [7] .

Директория **tools** содержит инструменты для сборки из исходных кодов, например, утилиту **nmake**. Ознакомьтесь с содержимым данной директории.

Ознакомление с базовой структурой исходных кодов ядра

Директория **base\ntos** содержит исходные коды ядра NTOS.

Структура этой директории изображена на рис. 39.3 .

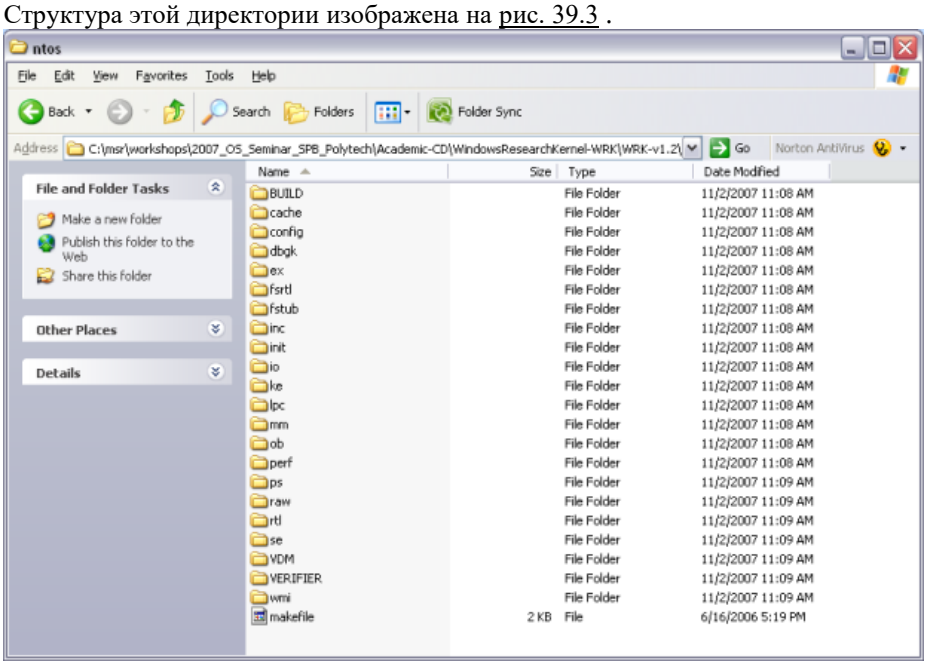


Рис. 39.3. Структура директории NTOS – базовой директории с исходными кодами ядра Windows

Первичные компоненты исходных кодов NTOS, включенные в пакет WRK, организованы следующим образом:

- **cache** - менеджер управления кэш-памятью
- **config** - реализация рестра
- **dbgk** - поддержка отладки в пользолвательском режиме

- **ex** - функции executive (куча ядра, синхронизация, установка времени)
- **fsrtl** - поддержка времени выполнения для файловой системы
- **io** - планировщик, управление центральным процессором, низкоуровневая синхронизация
- **lpc** - реализация локальных вызовов процедур (рассмотрены в "Обзор архитектуры и возможностей систем Windows 2000/XP/2003/Vista/2008/7 " , "Системные механизмы Windows ")
- **mm** - менеджер виртуальной памяти
- **ob** - менеджер объектов ядра
- **ps** - поддержка процессов и потоков
- **se** - функции безопасности
- **wmi** - инструментирование для управления Windows
- **inc** - заголовочные файлы, используемые в NTOS
- **rtl** - поддержка ядра времени выполнения
- **init** - инициализация ядра

Например, файл **ob\obinit.c** содержит исходный код инициализации компоненты OB (управление внутренними объектами) ядра Windows.

Ознакомьтесь с его содержимым.

Мы закончили краткое ознакомление со структурой WRK.

Задание повышенной сложности: Экспериментальная сборка Windows из исходных кодов

Для тех, кто уже имеет опыт сборки больших проектов и использования утилиты **make**, а также хорошо владеет техническим английским языком, предлагаем более сложное задание – сборку ядра Windows из исходных кодов.

Подробная инструкция по сборке дана в файле README базовой директории.

Приводим ее ниже:

Инструкция по сборке Windows

Building/deploying a WRK kernel for x86 [or amd64]

0. Copy the WRK into a directory, say %wrk%.
1. set arch=x86 [or amd64]
2. path %wrk%\tools\%arch%;%path%
3. cd %wrk%\base\ntos
4. nmake -nologo %arch%=
   will produce kernel files in BUILD\EXE\%arch%
   [wrkx86.* or wrkx64.*]
5. copy the kernel to %SystemRoot%\system32\
6. if x86, find the Multi-processor version of hal.dll [see below]
7. add a line to C:\boot.ini of the target system
   to boot this kernel and the MP hal [see below]
8. reboot and select the boot option for the new kernel
9. you will boot up on a kernel you built/linked yourself!
   [always keep the original boot.ini line and kernel/hal available so you
    can still boot your system if something fails with your WRK kernel modifications]
10. set up a debugger [see below]

Multi-processor hal (x86 only, amd64 hals are all MP)
   All hals are renamed hal.dll, so you have to use the link command to
   see what type of hal hal.dll really is:
      link -dump -all hal.dll | findstr pdb
   The MP hals have an 'm' in the native name of the hal, e.g. halmacpi.dll
   You may already have an MP hal installed on UP systems, due to hyperthreading.
   If the hal isn't MP, you need to find the MP hal that corresponds to the current hal
   the target system does have, i.e.
      halacpi.dll -> halacpim.dll   ; ACPI PIC-based PC [used by VirtualPC]
      halaacpi.dll -> halmacpi.dll   ; ACPI APIC-based PC
      halapic.dll -> halmps.dll   ; MPS
   Look in the WRK WS03SP1HALS\x86 directory for the MP hal you need.

Boot.ini
   Edit boot.ini (you may have to use attrib -h -s -r first)
   Copy the line for the first operating system listed to the end of the file and edit it.
      [boot loader]
      timeout=30
      default=multi(0)disk(0)rdisk(0)partition(2)\WINDOWS

[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="Windows Server 2003, Standard"
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="test" /kernel=wrkx86.exe /hal=halmacpi.dll

Note that the filenames must be short (8.3) names.
You can add additional options for debugging (as specified in the WinDbg/KD help).

Debugging WRK

The WinDBG/KD debuggers will work with the WRK. The documentation is pretty thorough, and includes information about how to debug across a serial port, locally (examining kernel data from user-mode), and debugging kernels running on VirtualPC.

Version 6.6.3.5 of the WinDBG/KD debuggers is available with the Curriculum Resource Kit Tools ("CurriculumResourceKit-CRK\CRKTools\Debugging Tools" directory on the CD). The latest version of the Windows Debugging Tools can be downloaded from http://www.microsoft.com/whdc/devtools/debugging.

Пояснения к инструкции по сборке Windows

HAL (Hardware Abstraction Layer) – компонента ядра Windows, реализующая уровень абстрагирования от аппаратуры. Использование подобной компоненты – многолетняя традиция Microsoft. Данный метод применен также в Microsoft Office и в академической версии .NET (Rotor).

Curriculum Resource Kit (CRK) – еще одна обучающая компонента для изучения ядра Windows, также предоставляемая фирмой Microsoft свободно для целей изучения и исследований. Ее структура и ссылка на ее дистрибутив описаны в "Академическая программа Microsoft Shared Source Initiative. Открытое ядро Windows для изучения и исследований (Windows Research Kernel)" . CRK содержит набор утилит, позволяющих иследовать различные особенности работы ОС. Необходимые пояснения даны также в книге [7] .

CD (компакт-диск), упоминаемый в инструкции, распространялся в 2006 – 2007 гг. фирмой Microsoft среди преподавателей университетов.

В частности, такой диск имеется и у автора курса.

Структура академического диска приведена на рис. 39.4 .

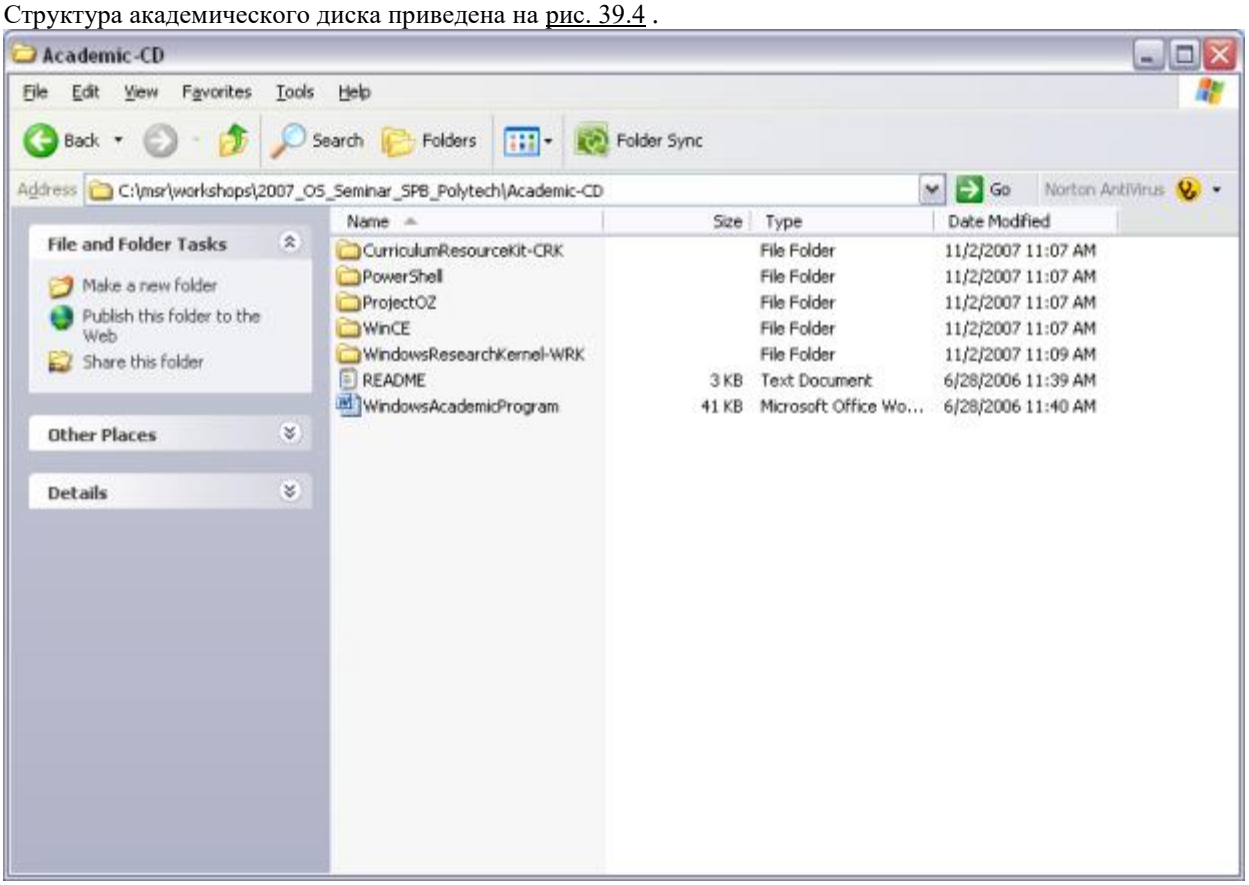


Рис. 39.4. Структура академического диска Microsoft с материалами для изучения

Контрольные вопросы ПЗ5 (ПК-1):

1. Что такое сегментная организация памяти?(ПК-1).
2. Что такое сегмент?(ПК-1).
3. Приведите примеры модулей кода и данных, память для которых распределяется в виде отдельных сегментов.(ПК-1).

4. Какую структуру имеет логический адрес при сегментной организации памяти?(ПК-1).
5. Что такое таблица сегментов?(ПК-1).
6. Какая информация хранится в элементе таблицы сегментов?(ПК-1).
7. Что такое регистр базы таблицы сегментов?(ПК-1).
8. Что такое регистр длины таблицы сегментов?(ПК-1).
9. На какой фазе (во время загрузки или исполнения) осуществляется перемещение при сегментной организации?(ПК-1).
10. Какие стратегии распределения памяти применяются при сегментной организации?(ПК-1).
11. Какие признаки защиты хранятся в элементе таблицы сегментов?(ПК-1).
12. Какое условие для номера сегмента проверяется при адресации?(ПК-1).
13. Что такое сегментно-страничная организация и для какой цели она используется
14. Что такое виртуальная память?(ПК-1).
15. Какие преимущества дает применение метода виртуальной памяти?(ПК-1).
16. Какие два способа используются для организации виртуальной памяти?(ПК-1).
17. Что такое страничная организация по требованию?(ПК-1).
18. Что такое сегментная организация по требованию?(ПК-1).
19. Что такое отказ страницы (page fault) и как ОС обрабатывает эту ситуацию?(ПК-1).
20. Что такое бит valid-invalid?(ПК-1).
21. Какие действия выполняет ОС при отсутствии свободного фрейма при обработке отказа страницы?(ПК-1).
22. Что такое эффективное время доступа к странице и как оно вычисляется?(ПК-1).
23. Что такое копирование при записи (copy-on-write)?(ПК-1).
24. Что такое файл, отображаемый в память?(ПК-1).

ПРАКТИЧЕСКАЯ РАБОТА № 6.

Изучение системы Linux

Целью лабораторной работы является практическое освоение операционной системы Linux – ее графической оболочки, входа и выхода, структуры рабочего стола, основных действий и настроек при работе в системе. Необходимый общий теоретический материал по архитектуре и особенностям ОС Linux представлен в "Обзор архитектуры и возможностей системы Linux: архитектура, ядро, распространение и лицензирование, принципы проектирования, управление процессами" и "Обзор архитектуры и возможностей системы Linux: управление памятью, ресурсами, файловые системы, драйверы устройств, сети, безопасность" данного курса.

Содержание

Аппаратура и программные инструменты, необходимые для лабораторной работы

Настольный или портативный компьютер с операционной системой Linux. В тексте и иллюстрациях данной лабораторной работы использована ОС Linux Red Hat.

Продолжительность лабораторной работы

2 академических часа

Обзор Linux

Linux – операционная система типа UNIX с открытым ядром, используемая главным образом как серверная ОС на многих аппаратных платформах.

В системе Linux доступны все классические инструменты и утилиты UNIX – grep (текстовый поиск), sed (потоковый редактор), awk (язык для потокового редактирования), make (утилита для сборки проектов, lex (генератор лексических анализаторов), yacc (генератор синтаксических анализаторов) и т.д. и, кроме этого, многие другие более современные инструменты и языки программирования.

Основные графические оболочки, используемые для системы Linux, - KDE и GNOME.

В данной лабораторной работе используется Linux Red Hat с графической оболочкой GNOME.

Запуск системы

Включите компьютер с инсталлированной ОС Linux.

При загрузке Linux (по сравнению с загрузкой Windows) на экран выдается значительное число текстовых сообщений, по которым Вы можете сориентироваться, на какой стадии находится загрузка ОС – какие драйверы, процессы демоны и другие компоненты загружаются в данный момент.

Наконец, примерно через 1-2 минуты загружается графическая оболочка, и Вам предлагается войти в систему, указав имя пользователя и пароль.

Вход в систему и аутентификация пользователя

Введите Ваше имя пользователя, затем пароль.

После входа в систему на экране визуализируется рабочий стол (рис. 40.1):

Рис. 40.1. Рабочий стол Linux

Структура рабочего стола

Рабочий стол несколько отличается от рабочего стола Windows (см. лабораторные работы 1 и 2). Фон по умолчанию – синий, на рабочем столе только Ваша домашняя директория, имеющая в Linux имя /home/user, где user – Ваше имя пользователя; стартовая точка для конфигурирования системы (Start here) и корзина (Trash). В нижней части - панель задач (taskbar) светло-серого цвета. Роль кнопки Пуск в Linux Red Hat играет красная шляпа (именно так переводится сочетание red hat) в левом нижнем углу. Кликнув на красную шляпу, пользователь может выбрать начальное действие: Accessories – стандартные программы, Games – игры, Graphics – графика, Internet – Интернет, Office – вызов офисного приложения OpenOffice, аналога Microsoft Office; Preferences – настройки, Programming – инструменты программирования (например, интегрированная среда GNU Emacs); Sound and Video – приложения для работы с аудио- и видеофайлами, System Settinggs – системные настройки, System Tools – системные программы (инструменты), Home folder – переход к Вашему домашнему каталогу (/home/user), Network Servers – настройка сетевых серверов, Run program – исполнение программ, Lock screen – защита экрана от входа других пользователей в систему, Log out – выход из системы, перезапуск или выключение компьютера.

Поэкспериментируйте со всеми пунктами стартового меню.

Работа с домашней директорией

Кликнув два раза иконку домашней директории на рабочем столе, визуализируйте содержимое Вашей домашней директории /home/user (рис. 40.2) с помощью браузера nautilus, компоненты среды GNOME, аналогичной Windows Explorer в Windows:

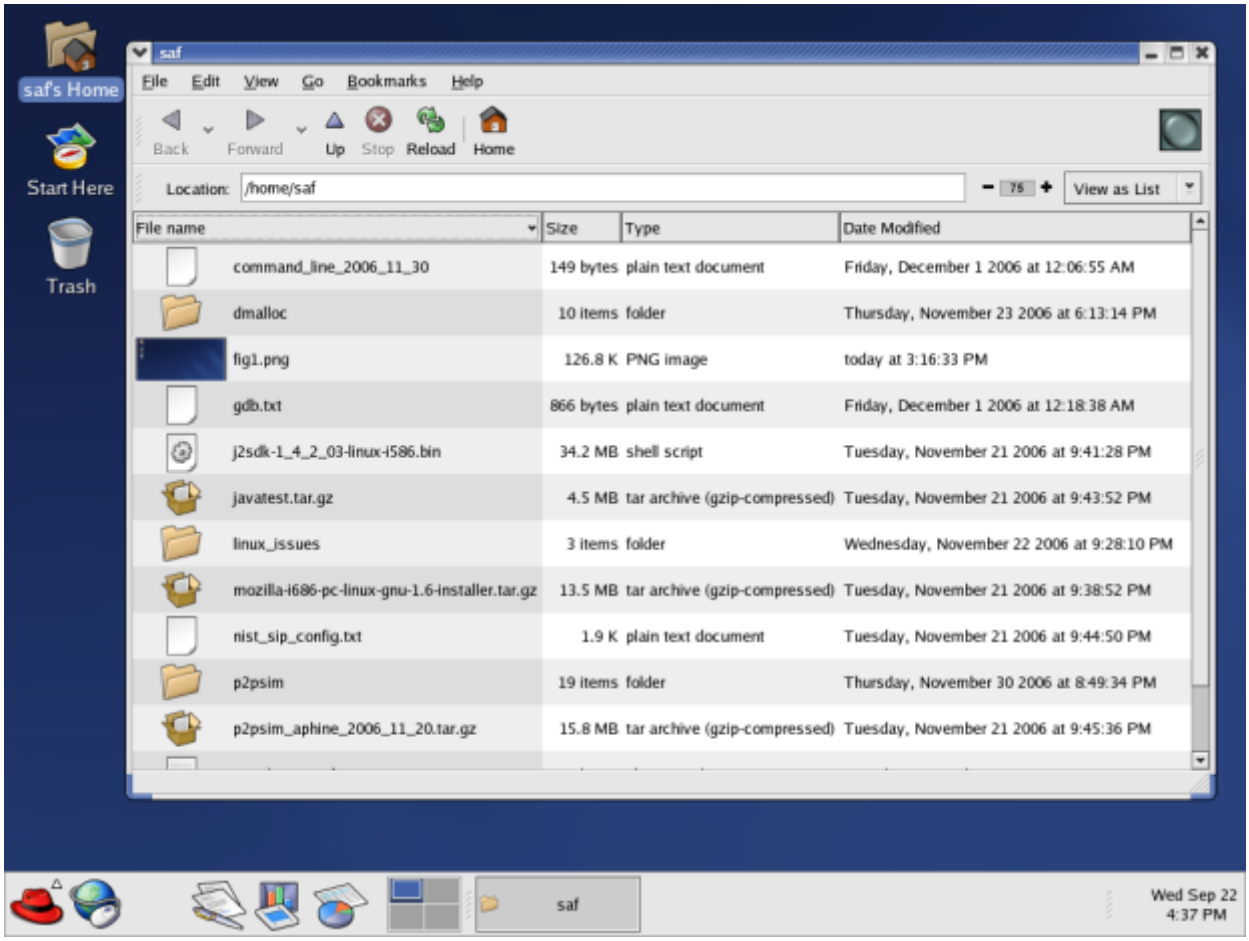


Рис. 40.2. Работа с директориями с помощью браузера nautilus

Вы видите почти привычные пункты меню, иконки и кнопки, например, Up (жирная серая стрелка вверх) – переход в родительскую директорию.

Выбор файла или папки в директории осуществляется одним кликом мышки, вход в директорию или открытие файла – двойным кликом мышки на имени директории или файла. При этом для файла выполняется действие его открытия, зависящее от его типа, - для текстовых файлов – вызов соответствующего редактора, для рисунков – их визуализация, для исполняемых кодов или командных файлов – запуск соответствующей программы или скрипта и т.д. Поэкспериментируйте на своем компьютере с навигацией по файлам и папкам и открытием файлов с документами.

Работа с папкой Start Here

Папка Start Here – аналог стартового меню.

Два раза кликнув на этой папке, перейдите в нее (рис. 40.3):

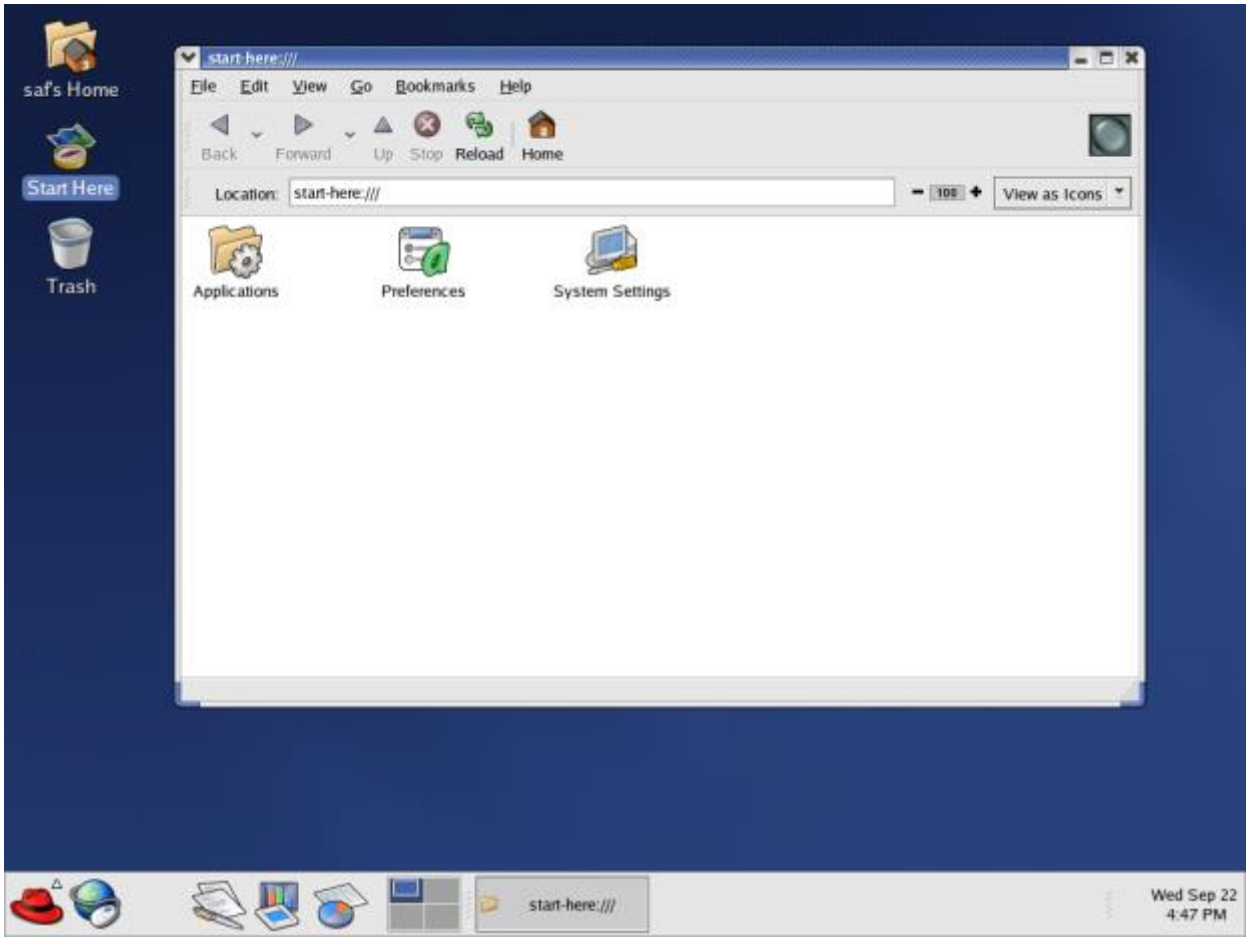


Рис. 40.3. Содержимое стартовой папки Start Here

Здесь Applications – запуск приложений, Preferences –настройки рабочего стола, System settings – системные настройки (аналог панели управления в Windows).

Дважды кликнув на Applications, перейди те в папку приложений (рис. 40.4):

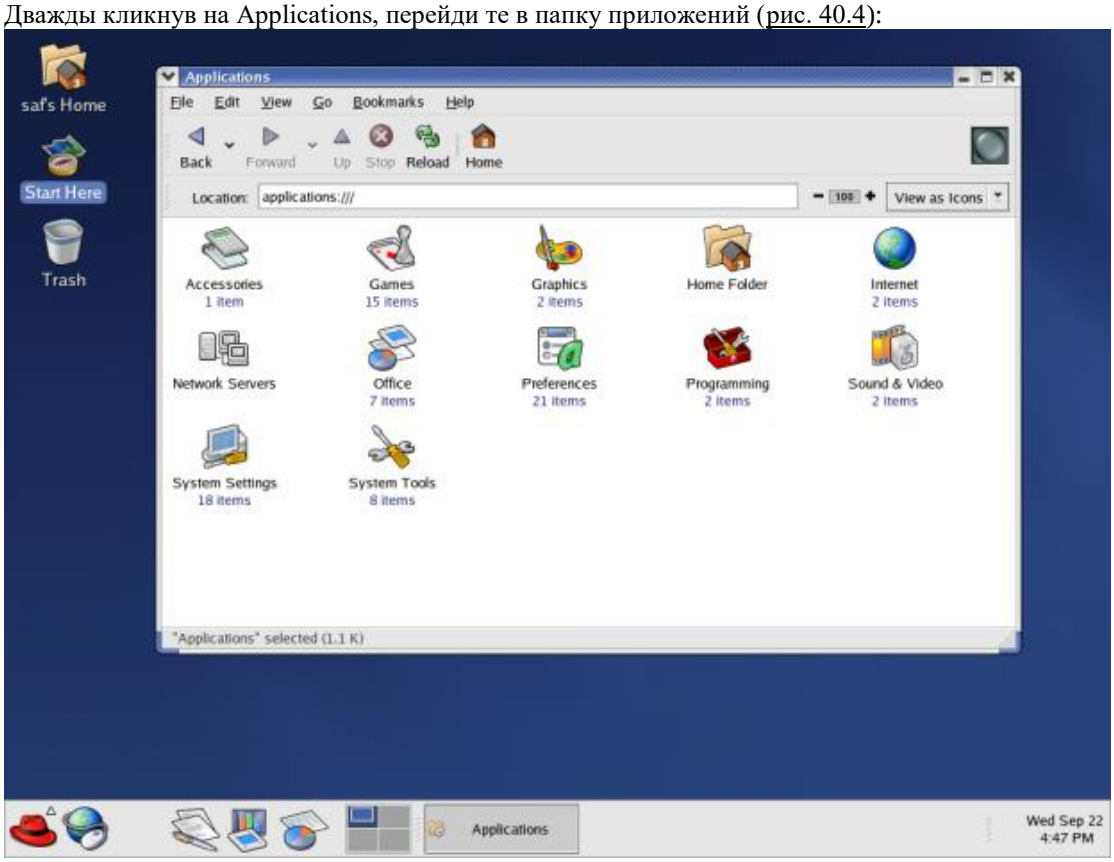


Рис. 40.4. Содержимое папки приложений в Linux

Приложения распределены по категориям – игры, офисные приложения и др. Для их запуска выберите нужную категорию и т.д. Теперь вернитесь в папку Start here (нажмите "Back" – стрелку влево), затем – кликните дважды папку Preferences (рис. 40.5):

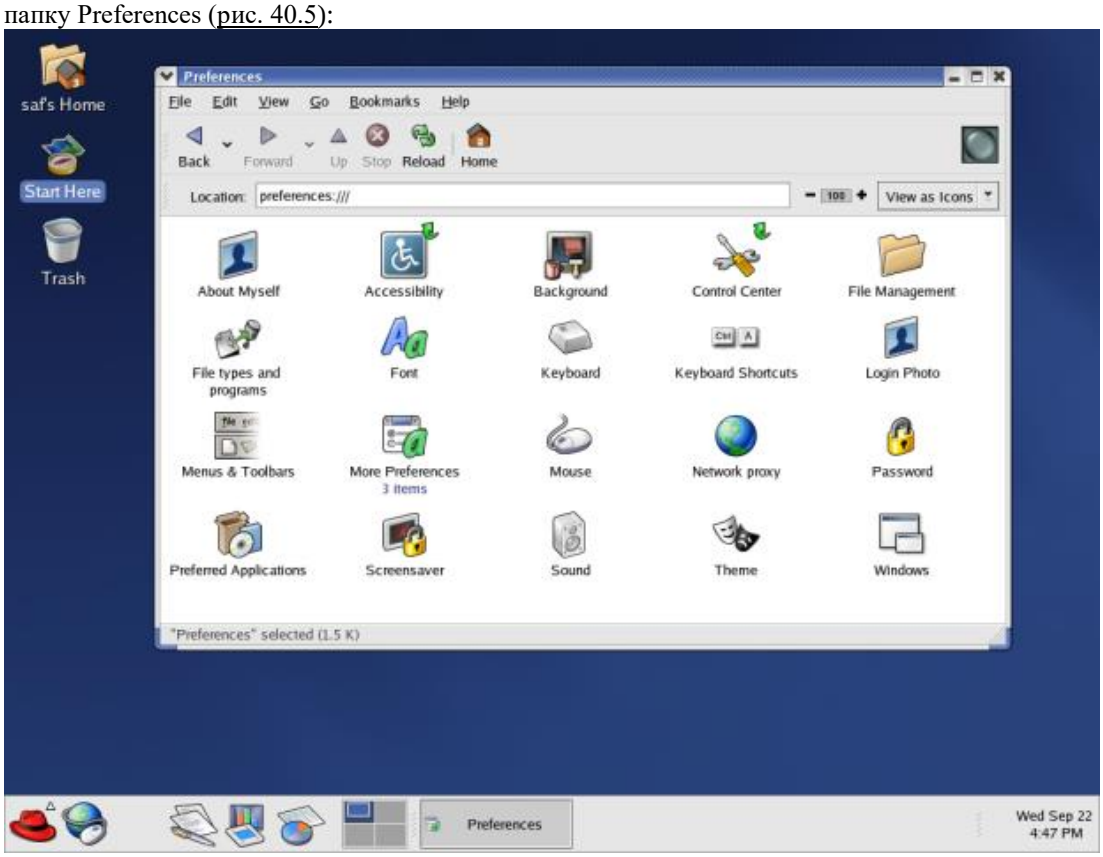


Рис. 40.5. Папка Preferences в Linux

Здесь Вы можете ввести сведения о себе, изменить настройки рабочего стола, мыши, звука, цветовых тем и др.

Поэкспериментируйте с пунктами этого меню (папки).

Затем вернитесь назад и войдите в папку System Settings (рис. 40.6):

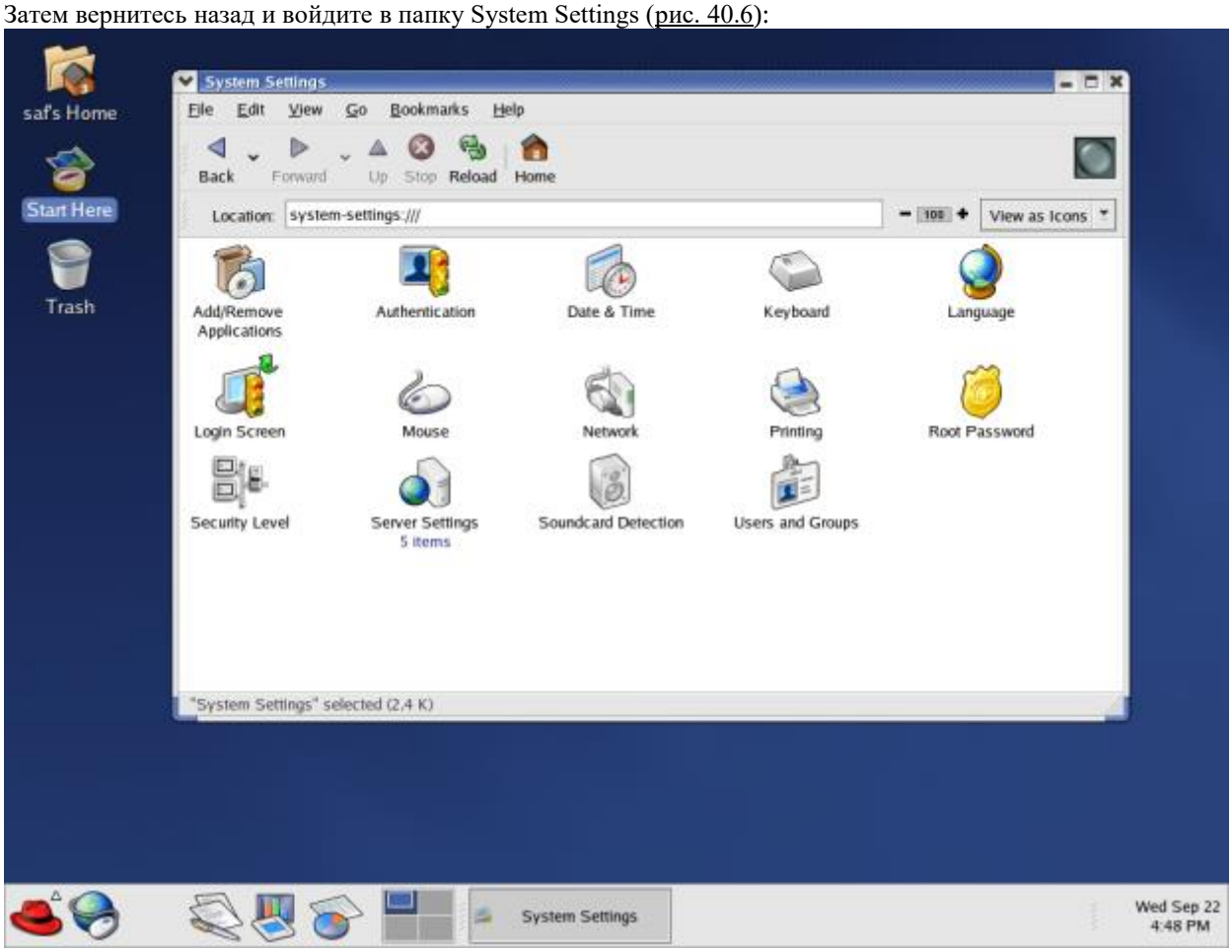


Рис. 40.6. Содержимое папки System Settings в Linux

Здесь Вы можете ввести нового пользователя, настроить серверы, задать пароль системного администратора (root) и др.

Работа из командной строки. Утилита Terminal

Специфика ОС Linux, как и ОС UNIX, в том, что любые системные настройки или другие действия в системе удобно выполнять не только в графической оболочке, но и из командного языка (shell). "Фанаты" UNIX и Linux предпочитают набирать и использовать сложные командные файлы – скрипты, нежели работать в графической оболочке. Изучим и мы "азы" работы в командном языке в Linux.

Командный язык в Linux доступен с помощью утилиты Terminal, которая запускается из стартового меню (красной шляпы) как **Красная шляпа / System tools / Terminal**. В результате визуализируется окно утилиты Terminal (рис. 40.7):

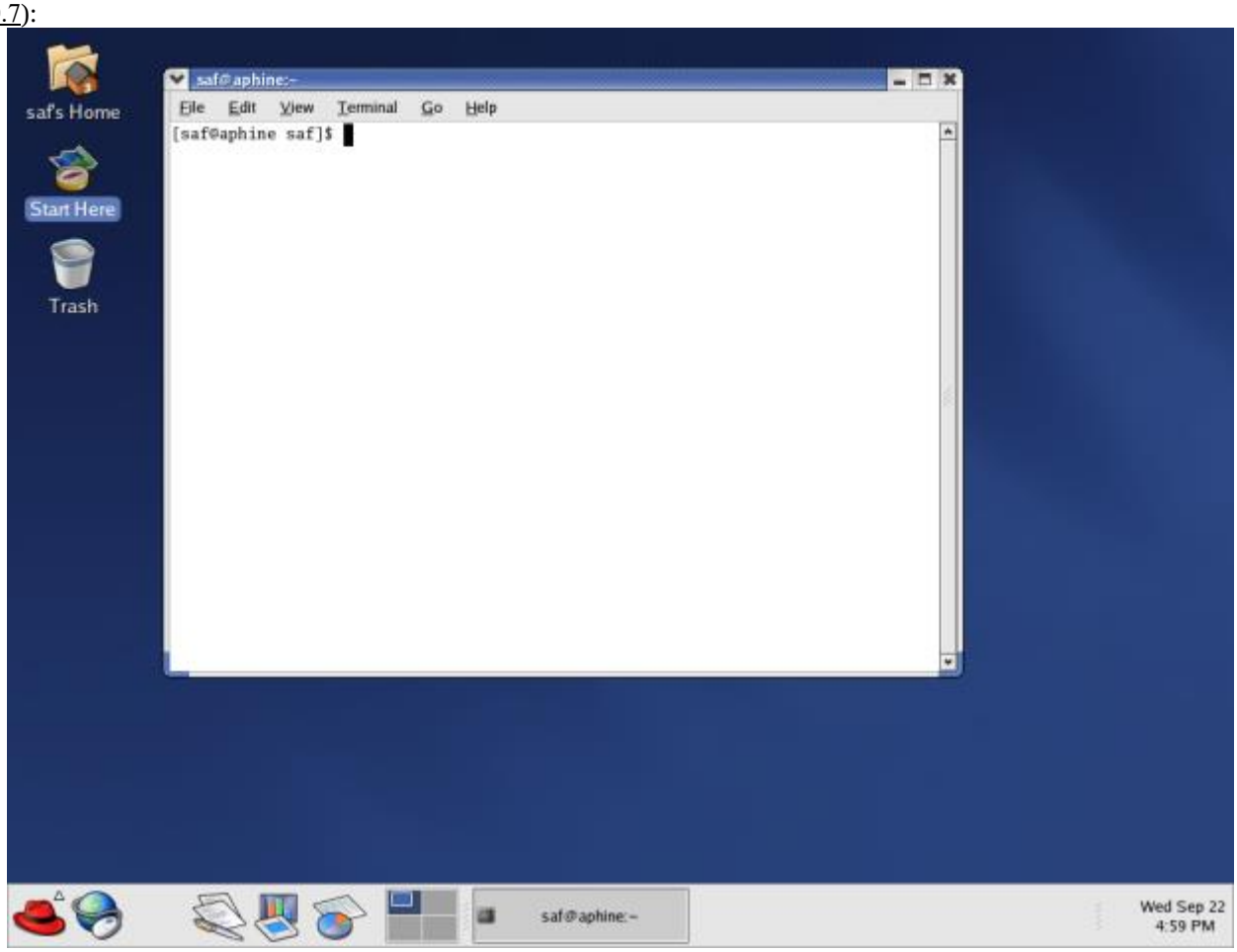


Рис. 40.7. Запуск утилиты Terminal для работы на командном языке
(Пуск / System tools / Terminal)

Вы видите приглашение (курсор) для ввода команды, включающее имя пользователя, имя компьютера и имя директории (в качестве начальной текущей директории устанавливается Ваша домашняя директория).

Наберите самую простую команду Linux (UNIX) – ls, вывод на экран содержимого директории. Результат выполнения команды для моей домашней директории показан на рис. 40.8 . Выполните команду ls и проанализируйте содержимое Вашей домашней директории:

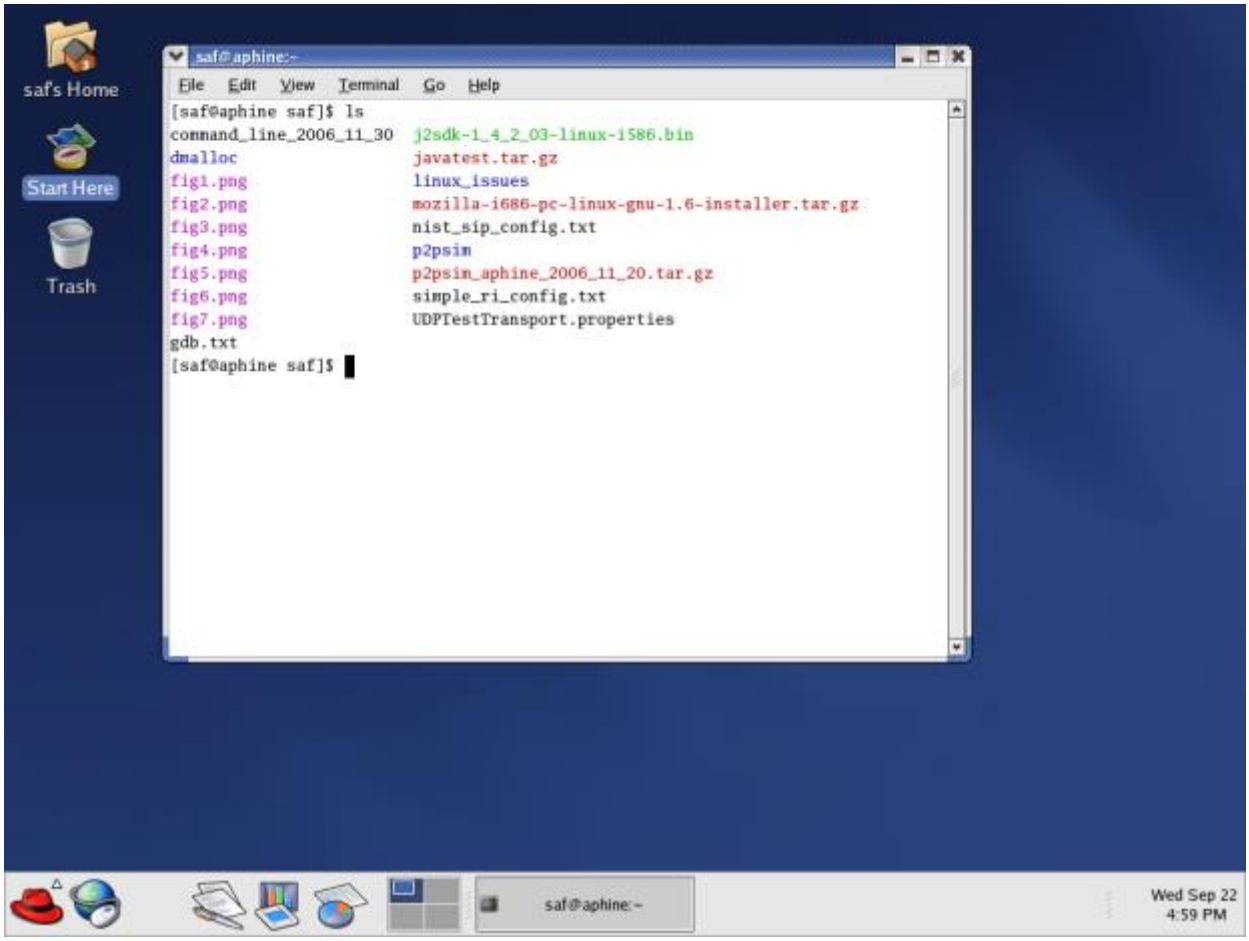


Рис. 40.8. Вывод содержимого домашней директории в Linux.

Теперь выполните еще несколько команд, классических для UNIX и Linux:

whoami ("кто я?") – выведется Ваше имя пользователя; если Вы вошли под именем системного администратора, то выведется root

uname ("UNIX name") – имя операционной системы; будет выведено Linux. Более подробная информация (версия ОС) выводится командой uname –a

ps -a – информация о процессах, запущенных в системе

ps –a | grep user – информация только о Ваших процессах в системе (где user- Ваше имя пользователя).

Соединение в сеть с Windows-компьютером. Сервер Samba.

Для подсоединения компьютера к локальной TCP/IP - сети необходимо выполнить для него сетевые установки – задать IP-адрес. Он задается обычно при инсталляции Linux.

Физическое подсоединение к сети сделайте (проверьте) путем подсоединения к сетевому разъему (RJ45) сетевого кабеля вида twisted pair (витая пара), который соединяет Ваш компьютер с сетевым концентратором (hub) или переключателем (switch). Наличие физического соединения индицируется зеленым световым индикатором (проверьте).

Для соединения в сеть служит сетевая карта (сетевой адаптер).

При использовании Linux, как правило, возникает проблема соединения в сеть с Windows-компьютером, таким образом, чтобы с Windows-машины были "видны" файлы указанной Вами директории Linux-машины (например, Вашей домашней директории).

Например, именно так автор курса подготавтивал данный текст лабораторной работы, получая "скриншоты" на Linux-машине, а затем пересылая их как .png-файлы по локальной сети на Windows-машину, используя Windows Commander.

Эта проблема решается с помощью серверного программного обеспечения Samba.

Все, что требуется, - это правильно задать настройки сервера Samba.

Для этого выберите: **Красная шляпа / System Settings / Server settings / Samba server**. Система попросит Вас ввести пароль системного администратора (root). Затем выполняется вход в папку, с помощью которой Вы сможете конфигурировать сервер Samba (рис. 40.9):

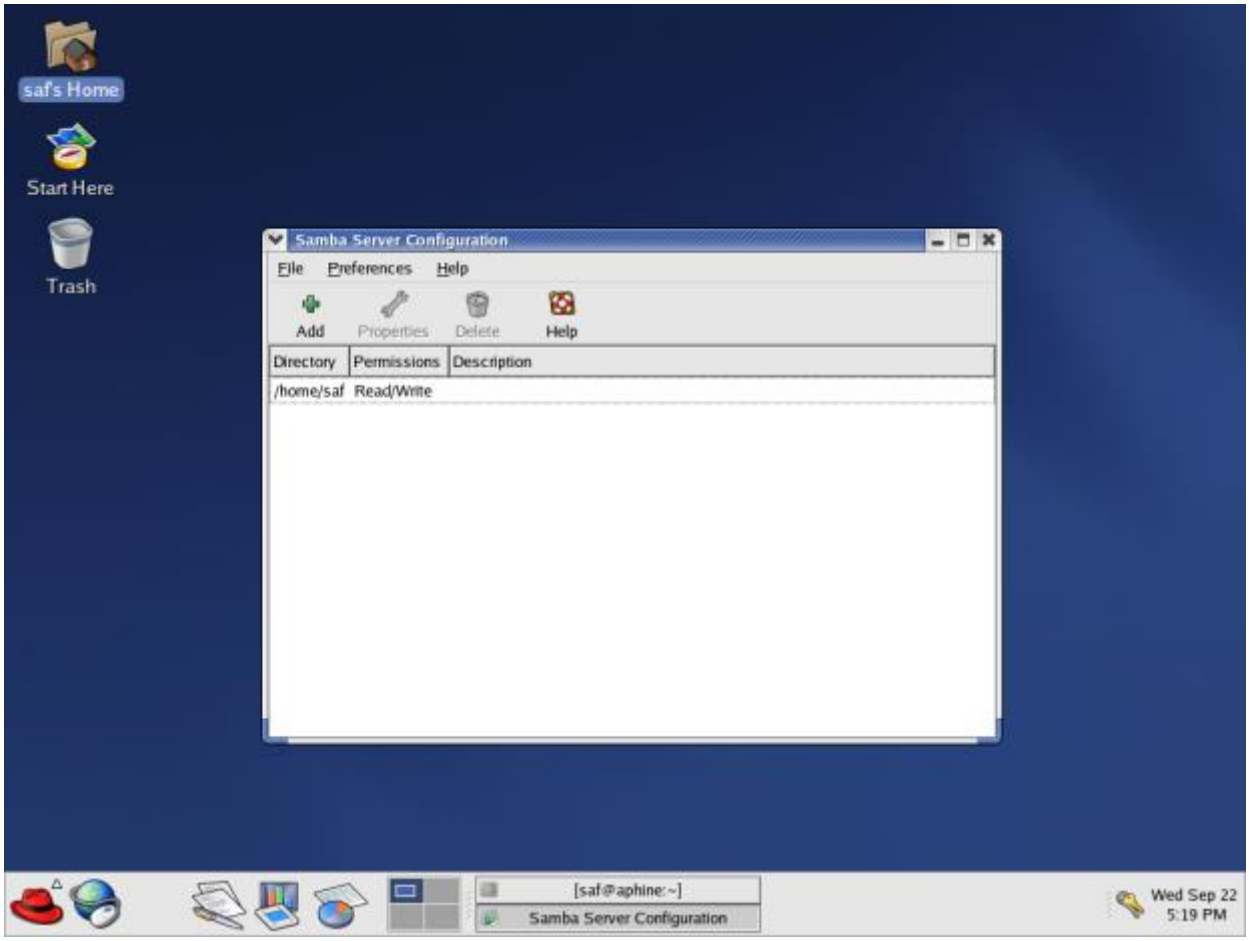


Рис. 40.9. Настройка сервера Samba.

На рис. 40.9 показан уже готовый результат – в конфигурацию сервера Samba (т.е. в список директорий, доступных на Windows-машине) включена домашняя директория с правами чтения и записи. Для этого сначала нажмите Add (зеленый плюс), выберите директорию для связи с Windows-машиной и задайте необходимые права доступа к ней.

Но это только половина работы. Теперь Вам необходимо внести в сетевую конфигурацию Вашего Linux-компьютера Windows-машину. Самый простой способ сделать это в системах UNIX или Linux – это отредактировать системный конфигурационный файл /etc/hosts, в котором задается список хостов (компьютеров) локальной сети – как Linux- , так и Windows-машин, с их IP-адресами. Данный файл может изменять только системный администратор.

Пусть имя windows-машины – aphrodite, IP-адрес – 192.168.205.13.

Для редактирования файла вызовите утилиту Terminal (см. п. 6). В ней выполните команду получения полномочий системного администратора: su (от **SuperUser**). Введите пароль администратора (root). Теперь Вы – администратор.

Выполните команду: vi /etc/hosts,

где vi – простейший редактор текстов, встроенный в UNIX и Linux.

Вы увидите начальное содержимое файла – имя компьютера localhost (локальная машина) с IP-адресом 127.0.0.1. Теперь, по образцу этой строки, введите аналогичную строку для компьютера **aphrodite** с его IP-адресом. Для этого переведите курсор на строку localhost и наберите команду редактора o (одну маленькую букву O). Введите новую строку для Windows-компьютера. Содержимое файла показано на рис. 40.10:

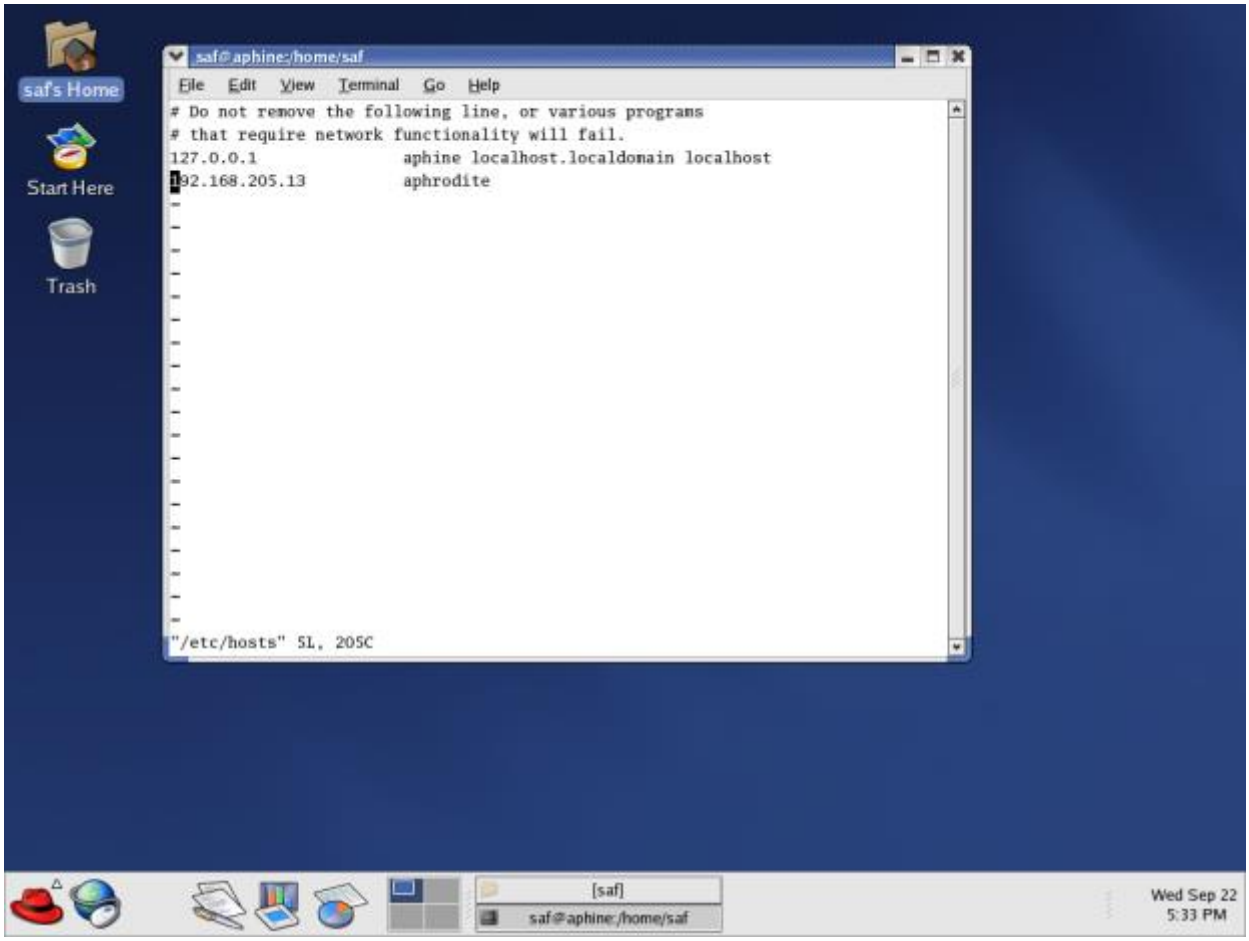


Рис. 40.10. Изменение сетевой конфигурации с помощью файла /etc/hosts

Теперь наберите <Esc>:wq (двоеточие в начале обязательно). Это команды записи файла и выхода из редактора. Теперь Ваша система "знает" сетевой адрес Windows-машины. Аналогично Вы можете ввести и сетевой адрес другой Linux-машины.

Выход из сеанса администратора – exit.

Сервер Samba готов к работе.

Теперь на Windows-машине, пользуясь утилитами Windows Explorer или Windows Commander, Вы можете увидеть, как элемент локальной сети, Вашу Linux-машину, а в ней – Вашу домашнюю директорию, доступ к которой по сети Вы предоставили.

Перешлите какой-либо файл с Linux-машины на Windows-машину и убедитесь, что пересылка прошла успешно.

Работа на удаленных компьютерах

С Вашей Linux-машины Вы можете удаленно войти по локальной сети на другую Linux-машину, запустив утилит командой rlogin linux2, где linux2 – имя другой Linux-машины. Введите Ваши имя пользователя и пароль на удаленной машине, и Вы попадаете в режим выполнения команд (утилиту Terminal) на удаленной Linux-машине. Поэкспериментируйте с выполнением команд на удаленной машине.

Аналогичный вход с Linux-машины на Windows-машину обычными штатными средствами невозможен.

Для работы в сети с файлами Windows-машины используйте сервер Samba, как объяснено выше.

Выход с удаленной Linux-машины – exit.

Выход из системы

Для выхода из Вашего сеанса пользователя выберите **Красная шляпа / Log Out / Log Out** – это выход из Вашего сеанса пользователя.

При выборе **Красная шляпа / Log Out / Shut Down** произойдет выход из Вашего сеанса пользователя, затем – выгрузка ОС и выключение компьютера.

В данной лабораторной работе Вы познакомились лишь с некоторыми базовыми возможностями ОС Linux. Более подробно с ними можно познакомиться в книге [9] .

Контрольные вопросы ПЗ6 (ПК-1):

1. Что такое файл?(ПК-1).
2. Какого типа информация может храниться в файле?(ПК-1).
3. Какую структуру может иметь файл?(ПК-1).
4. Какие программы интерпретируют содержимое файла?(ПК-1).
5. Каковы основные атрибуты файла?(ПК-1).

6. Каковы основные операции над файлом?(ПК-1).
7. Каким образом система определяет тип файла?(ПК-1).
8. Какие расширения имен используются в операционных системах?(ПК-1).
9. Какие методы доступа к файлам Вам известны?(ПК-1).
10. Какие операции определены над файлами прямого доступа?(ПК-1).
11. Какие операции определены над файлами последовательного доступа?(ПК-1).
12. Что такое индексный файл и для чего он используется?(ПК-1).
13. Что такое директория?(ПК-1).
14. Что такое раздел?(ПК-1).
15. Каковы основные операции над директорией?(ПК-1).
16. Каковы цели логической организации директорий?(ПК-1).
17. Какая организация директорий является наиболее предпочтительной и почему?ПК-1).
18. Что такое монтирование файловых систем?(ПК-1).
19. Что такое точка монтирования?(ПК-1).
20. Что такое общий доступ к файлам и почему он необходим?(ПК-1).
21. Что такое NFS?(ПК-1).
22. Что такое защита файлов?(ПК-1).
23. Какие полномочия защиты и для каких пользователей рассматриваются в UNIX?(ПК-1).

ЛИТЕРАТУРА

1. Gagne G, Galvin P., Silbershatz A. Operating System Concepts John Wiley & Sons, 2001 (6th ed.)
2. Таненбаум Э Современные операционные системы Питер, 2007
3. Таненбаум А Компьютерные сети Питер, 2006
4. Бернстайн Ф, Цикритзис Д. Операционные системы М.: Мир, 1977
5. Кальп Б Администрирование Windows Vista БХВ, 2008
6. Амарис К., Драуби О., Мистри Р, Моримото Р., Ноэл М. Microsoft Windows Server 2008 Вильямс, 2008
7. Руссинович М, Соломон Д. Внутреннее устройство MS Windows: Windows Server 2003, Windows XP и Windows 2000. 4-е изд Русская редакция (Microsoft Press), 2008
8. Майерс С MacOS X 10.5 Leopard БХВ, 2008
9. Баррет Д. Дж Linux: Основные команды Кудиц-Пресс, 2008
10. Владимир Сафонов Операционные системы и сети. Материалы курса
11. Престон Г, Робби А. Windows XP: Сборник рецептов СПб.: Питер, 200
12. Мак-Федрис П Microsoft Windows Vista. Полное руководство М.: Вильямс, 2007
13. Амарис К, Драуби О., Мистри Р., Моримото Р., Ноэл М. Windows Server 2008 R2. Полное руководство М.: Вильямс, 2010
14. Сайт проекта Parallel Dwafs М.: Вильямс, 2010
15. Климов А Программирование для мобильных устройств под управлением Windows Mobile Питер, 2009
16. Введение в платформу Windows Azure Питер, 2009
17. Вводная статья по Windows Azure на русском языке сентябрь 2010 г
18. Web-сайт системы Windows Live
19. Исходные коды Windows Research Kernel сентябрь 2010 г
20. V.O. Safonov Operating Systems and Networking. University undergraduate course
21. Hoare C.A.R Monitors — an operating systems structuring concept Communications of the ACM, 1974
22. Howard M., LeBlanc D Writing Secure Code Microsoft Press, 2003