

[ПРОГРАММИСТ]

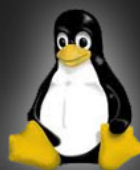
Программирование и алгоритмизация.
Для молодых и активных людей..

Алгоритм поиска
пути

Работа с графикой
на канве в Delphi

Создание
спектрограммы в
Delphi

Linux



Взаимодействие с сетевыми интерфейсами

LPT порт.
Компьютер в
роли
управляющего
контроллера

Передача звука по
сети. Прототип
VoIP телефона

И многое
другое...

Издается с марта 2010. Выходит ежемесячно
№3, май 2010 г.

Редакция:
Utkin, JTG, Алексей Шульга, Ксения Павлова,
Антон Бердников, Егор Горохов, Сергей Бадло

Дизайн и верстка:
Егор Горохов, Сергей Бадло

Авторский состав:
Utkin, Владимир Дегтярь, Александр Уколов,
Антон Бердников, Александр Терлецкий,
Олег Кутков, Егор Горохов

Контакты:
Авторские статьи направляйте на
maindatacentr@gmail.com
Вопросы и предложения для редакции
reddatacentr@gmail.com

Информационная поддержка:
Международная Академия Информатизации
(МАИН) РК www.academy.kz
Журнал «Радиолобитель»
www.radioliga.com
Клуб ПРОграммистов
www.programmersforum.ru

Примечание:
Издание некоммерческое. Все материалы,
товарные знаки, торговые марки и логотипы,
упомянутые в журнале, принадлежат их
владельцам. Статьи, поступающие в редакцию,
рецензируются. Мнение авторов не всегда
совпадает с мнением редакции. Перепечатка
материалов журнала и использование их в
любой форме, в том числе в электронных СМИ,
возможно только с разрешения редакции.
Тираж неограничен. Формат А4, 45 стр.

Учредитель:
Клуб ПРОграммистов
www.programmersforum.ru

Обложка номера:
Использован официальный талисман
Linux, пингвин Tux.

ТЕМА НОМЕРА

С новосельем с.0x02

НЕВЕРОЯТНО, НО ФАКТ

Любопытные факты с.0x03

LINUX ПРОГРАММИРОВАНИЕ. НАЧИНАЮЩИМ

Взаимодействие с сетевыми интерфейсами в Linux с.0x09

АЛГОРИТМЫ. ОБЩИЕ ВОПРОСЫ

Поиск пути с.0x0F

АЛГОРИТМЫ. ГРАФИКА В DELPHI

Как работать с графикой на канве в среде Дельфи. Урок 5-6 с.0x13

Создание спектрограммы в Дельфи с.0x17

ЛАБОРАТОРИЯ

LPT порт. Компьютер в роли контроллера. Часть 1 с.0x1C

Передача звука по сети. Прототип VoIP-телефона с.0x20

Разработка ресурса журнала. Часть 1 с.0x27

ЮМОР. ХОХМЫ ПРОГРАММИСТОВ

Фразеологизмы и байки с.0x2A

От редактора. Дорогие друзья! Добро пожаловать в майский выпуск журнала «ПРОграммист» от клуба программистов www.programmersforum.ru. Сегодня у нас праздник. Благодаря Василию Мединцеву, Алексею Шульге и Егору Горохову у журнала появился еще один небольшой, но уютный домик. Милости просим погостить в нашей резиденции <http://procoder.info>. Ну и какое-же новоселье без подарков. В Редакции пополнение ведущим рубрики новостей Антоном Бердниковым и женской половиной. Рады представить Вам Ксению, второго литредактора нашего журнала...



В этом выпуске жаждущих читателей порадует новым практическим материалом Владимир Дегтярь по использованию компьютера в качестве управляющего контроллера. Несомненно, не забыл он и о продолжении серии уроков по графике. Александр Терлецкий дебютирует со статьей по созданию спектрограммы для движка BASS. **Utkin** поможет найти кратчайший путь и выведет оптимальный маршрут. Олег Кутков статьей по особенностям взаимодействия с сетевыми интерфейсами не оставит равнодушными фанатов Linux-а. Рубрику «Лаборатория» на это раз поддержал Александр Уколов с материалом по практике приема и передачи звука по сети. Программный прототип VoIP телефона не за горами.

Еще одним подарком к новоселью стала информационная поддержка нашего журнала со стороны Международной Академии Информатизации МАИН РК www.akademy.kz и журналом «Радиолобитель» www.radioliga.com. Еще раз спасибо Василию www.kotoff.info и Вам, дорогие читатели, за ваш интерес к журналу и ценные советы!

Рубрики журнала (плавающие)

- Новости программирования
- Отдел тестирования
- Общие вопросы (вопросы правового использования, личные мнения и т.д.)
- Алгоритмы (без привязки к языку и платформе)
- Юмор (специфические хохмы программеров)
- Реализация (тонкости программирования)
- Разработка (создание программных проектов от этапа ТЗ до работающей программы)
- Переводные материалы
- Рубрика про железки / Лаборатория

Общие требования к материалам

У нас нет категоричных требований к оформлению, но в связи с особенностями верстки (используется свободное ПО «SCRIBUS») и облегчения труда редакторов, есть некоторый желательный минимум:

- статья должна иметь выраженную структуру с разделами и содержать – название статьи, сведения об авторах, экскурс или введение, сведения об используемых средствах разработки, теоретическую и/или практическую часть, заключение (выводы, чего добились) и ресурсы к статье (код, интернет-ресурсы, литература)
- текст статьи в формате MS Word, [VK WordPad](#) или обычным текстовым файлом, шрифт Arial 10
- все рисунки, таблицы должны иметь упоминание в тексте и иметь подпись
- рисунки к статье должны прилагаться **в виде отдельных файлов** в формате PNG, BMP или TIF
- разделы статьи отделять двумя <ENTER>
- не используйте табуляцию и лишние пробелы без необходимости
- по присланным материалам автор получает рецензию и корректирует статью согласно замечаниям
- шаблон для написания статьи можно взять [тут](#), бесплатный редактор VK WordPad можно взять [тут](#)

Замечание редактора обязательно должно вызывать реакцию автора. Если автор не согласен с комментарием или исправлением редактора, он должен разъяснить вопрос в виде комментария к статье на ящик-копилку. Если редактор согласен с тем, что ошибся, вопрос снимается. Если редактор не согласен, решает группа.

С уважением, Редакция

Месяц май выдался щедрым на события как в историческом плане, так и в современности. Такое множество происшествий мы не в состоянии охватить одной журнальной заметкой, но основные постараемся донести до вас, дорогие читатели. Итак...



Антон Бердников by Aexx

www.programmersforum.ru/member.php?u=55969

1 мая 1918 года родился Башир Искандерович Рамеев (1918-1994) – один из основоположников вычислительной техники в СССР, главный конструктор семейства вычислительных машин «Урал».



Автор первого в СССР проекта «Автоматическая цифровая электронная машина». В нем было описание принципиальной схемы машины, определены арифметические операции в двоичной системе счисления, управление работой машины осуществлялось от датчика, считывающего программу, записанную на перфоленту, и обеспечивающего выдачу результатов на такую же ленту. Авторское свидетельство от 4 декабря 1948 года на имя И.С. Брука и Б.И. Рамеева было первым в СССР зарегистрированным изобретением в области цифровой электронной вычислительной техники.



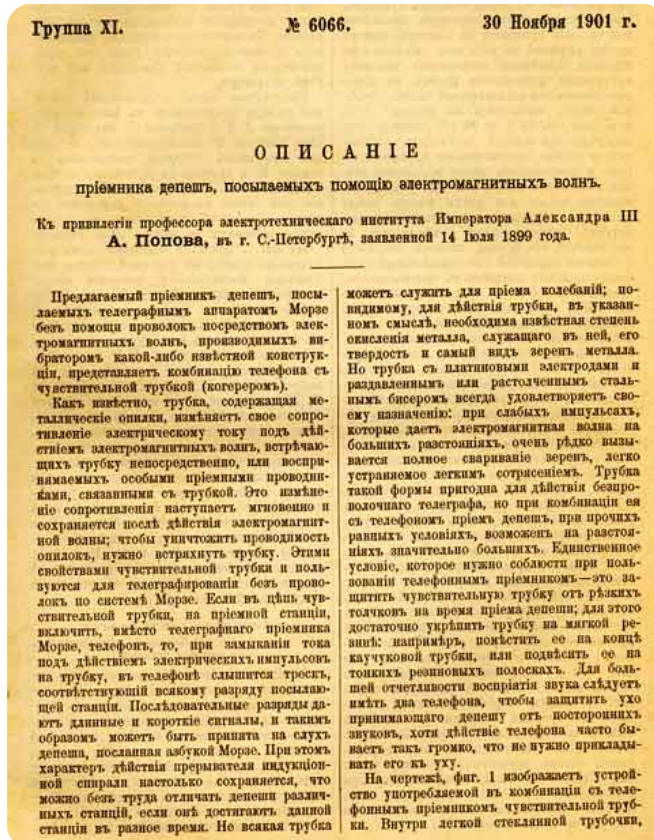
3 мая 1912 года родился Юрий Яковлевич Базилевский (1912-1983) – талантливый инженер-компьютерщик, главный конструктор ЭВМ «Стрела» и автоматизированного вычислительного комплекса приема и обработки радиолокационной информации для системы противовоздушной обороны «Даль-111».

3 мая 1984 года уроженец Хьюстона (штат Техас, США) Майкл Саул Делл зарегистрировал на свое имя компанию PCs Limited, Inc., арендовал однокомнатный офис и нанял сотрудника, который занялся администрированием и финансами. Так началась история Dell Computer Corporation (DCC) – фирмы, ставшей одним из крупнейших в мире производителей персональных компьютеров.

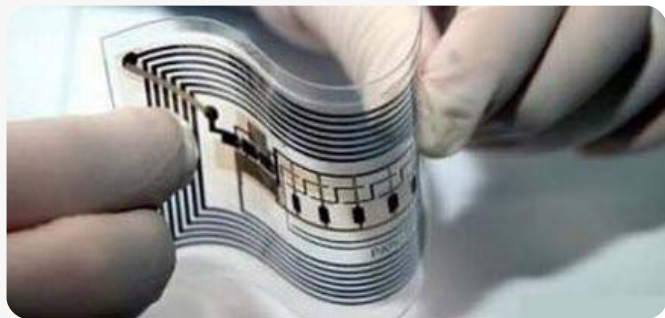
Надувные колонки как альтернативу привычным разработала компания Sebor. Приведение аксессуара в рабочее состояние занимает всего лишь пару минут. Колонки оснащены встроенным усилителем и регулятором громкости, выходная мощность их равна 10 Вт. Цена от 35 евро.



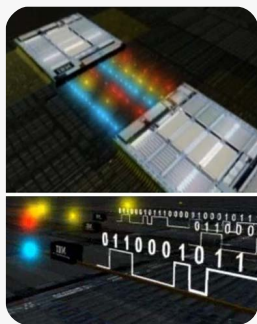
7 мая 1895 года Александр Степанович Попов на заседании физического отделения Российского физико-химического общества продемонстрировал свой грозоотметчик и прочитал доклад «Об отношении металлических порошков к электрическим колебаниям», здесь же он высказал мысль о возможности применения грозоотметчика для передачи сигналов на расстояние. 7 мая отмечается в нашей стране как День Радио.





Нанопечать RFID-меток возможно станет

альтернативой штрих-коду на товарах. До настоящего времени радиочастотные метки изготавливались на кремниевой основе. Использование пластика или бумаги в качестве исходного материала, а также поддержка «рулонной» печати поможет многократно сократить затраты на производство этих компонентов. Ключевым компонентом новой технологии являются особые чернила для струйного принтера, содержащие углеродные нанотрубки. Эти чернила используются для создания тонкопленочных транзисторов, которые в свою очередь являются основой для пассивных RFID-меток, содержащих до 16 бит информации и наносимых на бумагу или пластик методом печати.

**Нанофотонный лавинный фотодетектор** изобрели

ученые корпорация IBM, что делает возможным создание компьютерных чипов, в которых сигналы между транзисторами и ядрами будут передаваться при

помощи импульсов света. В основе работы нового фотодетектора лежат свойства полупроводника. При поступлении на устройство одного импульса света в его материале высвобождается несколько носителей зарядов, которые затем высвобождают другие носители, создается известный эффект лавины. Вспомните наши отечественные лавинные транзисторы ГТЗ38. Устройство способно принимать оптический сигнал со скоростью 40 Гбит/с и при минимальных помехах выдавать по 400 млрд бит в секунду цифровой информации при напряжении всего 1.5В.

Сенсорную кожу разработали ученые из

университета

Карнеги-Меллон

совместно с

компаний

Microsoft.

Технология

Skinput

позволяет



превратить кожу человека в подобие сенсорного экрана. Точнее говоря, проводя или нажимая пальцем по поверхности кожи ладони или предплечья, можно, например, контролировать звонки и писать сообщения, набирать телефонный номер, проигрывать музыку или играть хоть в



тетрис. Система может

реагировать даже на незаметные движения, такие как щипок или дерганье мышцы. Меню

проецируется на руку, а нажатия улавливает и интерпретирует специальный браслет с пьезоэлектрическими датчиками на верхней части руки.

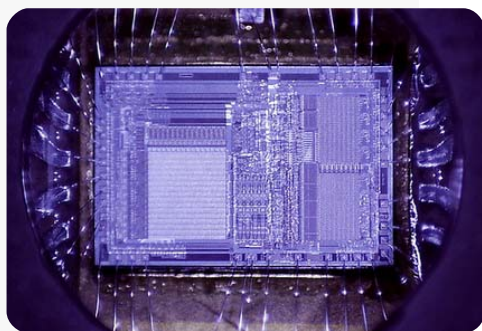
Телефон, читающий по губам

продемонстрировала на выставке CeBIT 2010 группа исследователей из технологического института города Карлсруэ (Германия). Технология позволит владельцам сотовых телефонов обмениваться информацией по телефону, не издавая ни единого звука.



Устройство использует методику электромиографии (применяется для диагностики некоторых нервных заболеваний), то есть измеряет электрическую активность движений мышц лица во время разговора. Крошечные электрические сигналы, производимые лицевыми мышцами, оно записывает даже тогда, когда человек вообще беззвучно воспроизводит слова. В настоящее время для снятия показаний используется 9 электродов, закрепляемых на коже, но в будущем ученые надеются избавиться от проводов и эти электроды будут встроены в мобильные устройства. Технология имеет огромное практическое значение: немые смогут говорить, говорящие перестанут орать в трубку. В метро или другом шумном месте микрофон телефона можно будет отключить и беседовать одними губами. Сейчас устройство понимает английский, французский и немецкий, а вот с языками вроде путунхуа может возникнуть проблема, ибо там важное значение имеет тон, который не передается движением лицевых мышц. На данный момент эффективность распознавания устройством человеческой мимики составляет 99%.

ISQ куплена за \$187,5 млн инвестиционным фондом Digital Sky Technologies, совладельцем которого является российский предприниматель Алишер Усманов. Digital Sky Technologies владеет контрольным пакетом акций Mail.ru (53,2%) и Astrum Online Entertainment, 30% социальной сети Vkontakte.ru, 25% в системе электронных платежей OE Investments, 100% портала HH.ru и 70% эстонской Forticom (основного владельца Odnoklassniki.ru), а также небольшой долей акций социальной сети Facebook (3,5%).



Спин атома кобальта смогли «сфотографировать» исследователи из университе-

та Огайо. Несмотря на то, что спин (собственный момент импульса квантовой частицы) является ключевым свойством квантовых частиц и используется в квантовых вычислениях уже десятки лет, его изображение представлено человечеству впервые. Физики использовали специально созданный сканирующий туннельный микроскоп, с помощью которого перемещали атомы кобальта по марганцевой подложке. Атомы кобальта при этом меняли свой спин и на изображениях четко видна зависимость высоты и формы пиков атомов от направления спинов. Исследования показывают, что ученые могут наблюдать и управлять спинами атомов, что может привести к созданию электроники атомных размеров и новым направлениям спинотроники. Однако до практического применения полученных результатов пока далеко. Для этого необходимо научиться получать необходимый эффект при комнатной температуре, а не охлаждать пластину марганца жидким гелием до 10° K.

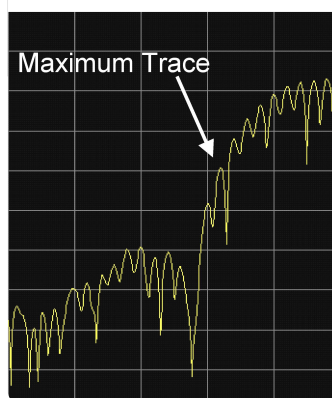


Новые модификации анализаторов спектра

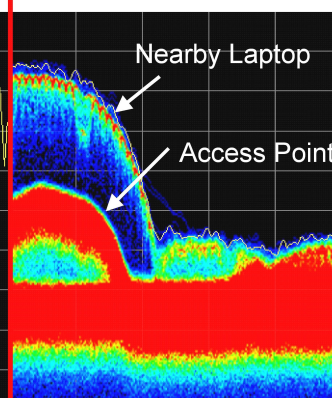
в реальном времени с технологией обработки и записи

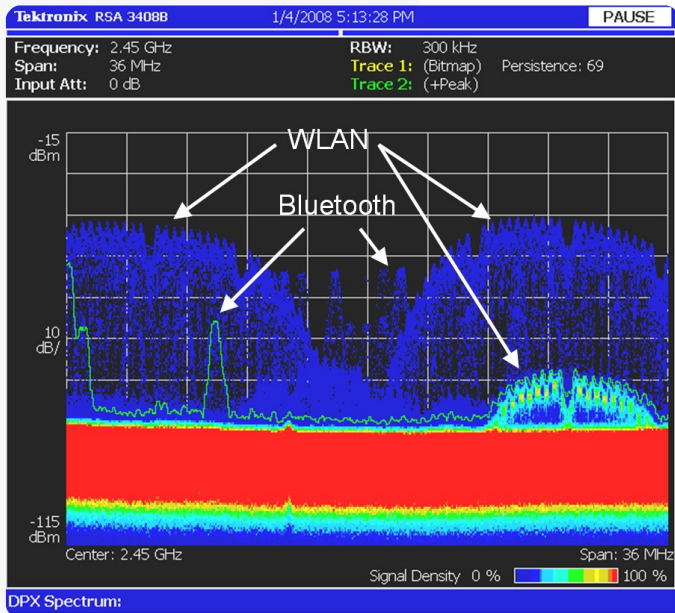
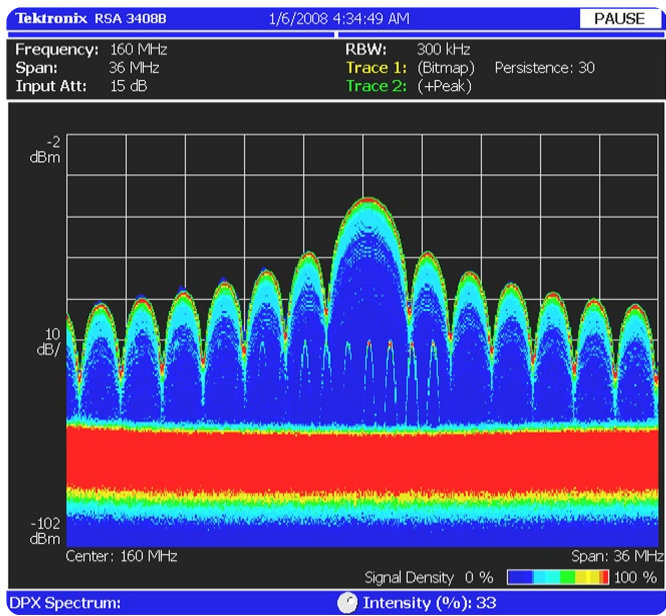
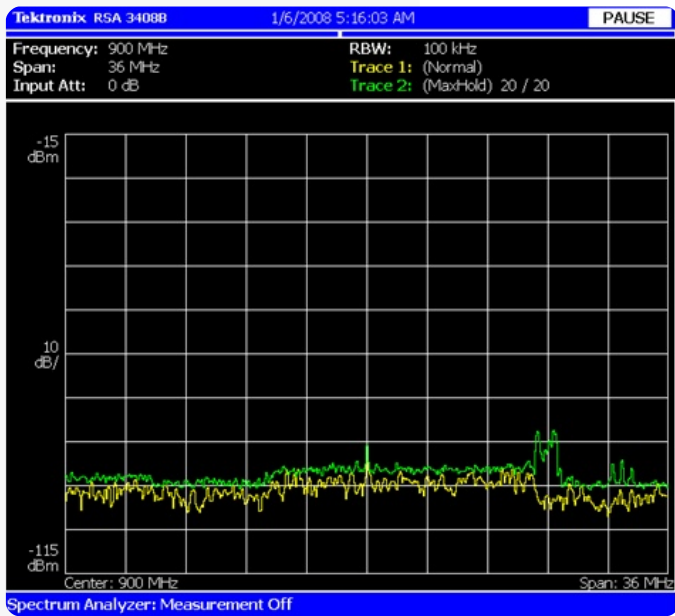
изображений сигнала DPX™ представила компания Tektronix Inc. Модели серий RSA3000B и RSA6100B поставляются в двух исполнениях с разными диапазонами перекрываемых частот: 0-3

Conventional Spectrum Analyzer Display

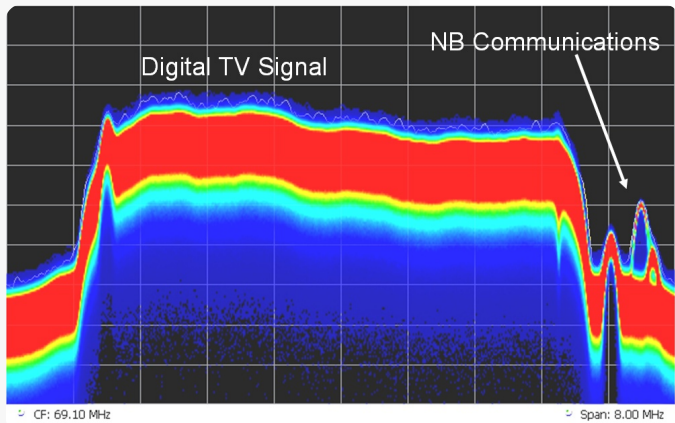


RSA 3000B DPX™ Spectral Display





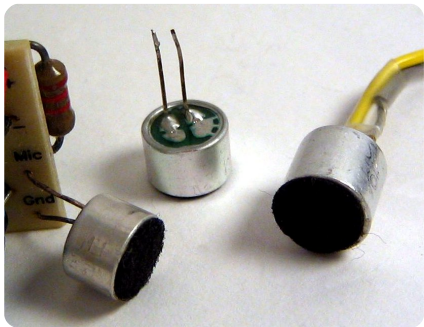
ГГц или 0-8 ГГц. Технология обработки изображений сигналов DPX, позволяет просматривать спектр в режиме реального времени, при скорости обработки более 48000 измерений спектра в секунду. Процессор обработки изображений сигналов с эффектом



послесвечения позволяет выявлять в динамических сигналах отклонения и частоту их повторения, а также обеспечивает мгновенную обратную связь при временных изменениях сигналов. Благодаря этому можно быстро просмотреть на экране переходные процессы и сигналы, которые раньше рассмотреть было невозможно, поскольку они маскировались другими сигналами, или для их выявления приходилось проводить автономный анализ, занимающий много времени. Основные виды анализируемых тестовых сигналов (W-CDMA, GSM/EDGE, CDMA, HSDPA, TD-SCDMA, WLAN 802.11a/b/g, HDTV и даже радиочастотных меток RFID) представлены на скриншотах выше, а видеотестирование вы можете просмотреть тут <http://raxp.radioliga.com/cnt/s.php?p=dpx.wmv>

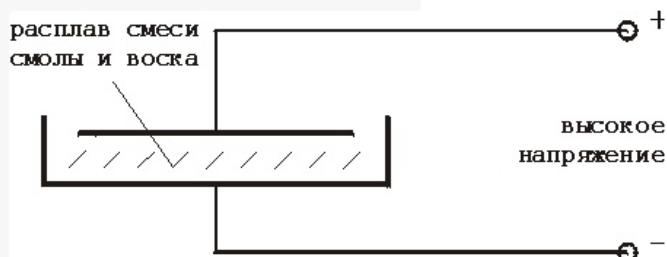
Этот удивительный электрет

Многим из нас знаком советский электретный микрофон типа МКЭ-3, а те, кто не знаком, все равно применяют его или его аналоги в повсеместной практике, даже не догадываясь о замечательных свойствах материала электрета (поляризованного диэлектрика), используемого в нем.



Электреты – это поляризованные диэлектрики, состоящие из жестких электрических диполей, которые в электрическом поле напряженностью около 10 кВ/см переводятся в аморфное твердое состояние и сохраняют поляризацию длительное время. Таким образом, электреты являются аналогом постоянного магнита, но обладают не магнитным полем, а электростатическим полем.

Не менее примечательна история появления этого материала... В 1943 году во время боевых действий на Тихом Океане американский флот захватил японский эсминец. Для изучения вражеской техники на корабль прибыли специалисты флота, осмотрели все – от трюма до капитанской рубки. Связист подробно изучил систему телефонной связи, она работала как часы. Одного он не мог понять, как система может работать без источника питания, ни батарей, ни аккумуляторов не было и в помине!



В конце-концов, физики разобрались в принципе работы телефонной связи на японском корабле. Ее работа стала возможной благодаря открытию японского физика Егучи, еще в 1922 году получившему новые материалы – электреты (с 1922 года работу с электретами Егучи проводил в обстановке строжайшей секретности для Министерства обороны Японии). Егучи получал электретные материалы из смеси смолы карнаубской пальмы и воска. Нагрев смесь до расплавленного состояния Егучи подал на нее напряжение 10 кВ, после застывания таблетка электрета сохраняла электростатический заряд

высокой напряженности в течение нескольких часов (см. рецептуры). Современные электреты могут сохранять заряд до 100 лет, величина заряда достигает $5 \cdot 10^{-8}$ Кл/см².

В настоящее время электреты получают из таких

материалов как: поли-тетрафторэтилен (поверхностный потенциал около 527 В), полиметилметакрилат или органическое стекло (3965 В), рутиловая керамика, смолы, воск, полимеры, титанаты щелочноземельных

металлов и даже растворы органических веществ в летучих растворителях / Под ред. Сергея Бадло

Рецептура N1

Нагреваем 2-5 г касторового масла до температуры 75-90 °С и высыпаем в него при помешивании 50 г мелкоизмельченной канифоли (если возится неохота, то возьмите и используйте парафин вместо канифоли и масла, расплавьте его и поместите между электродами – после остывания электрет готов). После чего расплав наливаем в плоскую баночку (стеклянную) или на предметное стекло и помещаем его между электродами, на которые подается напряжение от высоковольтного выпрямителя (5-10 кВ) или электростатической машины. Толщина слоя диэлектрика должна быть менее 4 мм. После остывания канифоль достается из баночки. Электрет готов.

Докажем, что электростатическое поле электрета имеет высокую напряженность. Если около неоновой лампочки быстро провести электретом, то она ярко вспыхнет, так как при пересечении лампой линий электростатического поля на электродах лампы наводится переменное высокое напряжение и лампа начинает светиться. Застывший диэлектрик – электрет способен сохранять заряд в течение нескольких суток. Хранить электреты можно завернутыми в алюминиевую фольгу.

Более стабильные электреты можно получить при нагреве диэлектриков до температуры меньшей или равной температуре плавления, а затем охлаждая их в сильном электрическом поле. При застывании органических растворов в сильном электрическом поле получают так называемые «криоэлектреты». Существуют и другие разновидности электретов: фотоэлектреты, трибоэлектреты и др.

Рецептура N2

Вместо канифоли и касторового масла можно между электродами в баночке поместить какой-нибудь полимер, например – капрон, растворенный в небольшом количестве растворителя. Подать высокое напряжение и после полного испарения растворителя – электрет готов.

Этой статьей я бы хотел открыть цикл публикаций, связанных с самым интересным и захватывающим в Linux - программировании. Я не собираюсь рассказывать о том, в чем писать программы, как их компилировать и запускать, информацию по данным вопросам найти очень легко и я думаю, что любой с этим справится. Языки программирования, которые будут использоваться в статьях: C, C++, bash script...



Олег Кутков

by OlegKutkov elenbert@gmail.com

В последнее время программисты и простые пользователи начинают проявлять все больший интерес к Unix-подобным операционным системам, в частности к Linux. К сожалению, новичкам, не всегда просто разобраться в новой среде, даже не смотря на то, что Linux считается одной из самых хорошо документированных ОС. Информация, как правило, разбросана по форумам, множеству отдельных статей и блогов. Основное содержимое данных материалов касается администрирования и настройки дистрибутивов, программистам же, как правило, приходится довольствоваться man-документацией или автоматической doxygen-документацией (документация, сгенерированная автоматически, на основе комментариев в исходном коде). К тому же, как это часто бывает - наиболее интересный материал на английском языке. Безусловно, данную ситуацию следует исправлять.

Начало. Общие сведения

Программировать в Linux очень просто и легко. Для программистов созданы практически идеальные условия: множество мощных инструментов, открытые исходные коды, сама организация системы, множество фреймворков. Работа с файлами, строками, массивами, классами, контейнерами, в Unix-среде, практически ничем не отличается от таковой в Windows, это касается и множества других стандартных функций и библиотек. Различия начинаются на более низком уровне. Предлагаю разобраться, как работать с сетевыми интерфейсами.

Как известно, сетевые интерфейсы Linux обозначаются короткими строковыми именами - eth0, wlan0, lo и т.д. Интерфейсу можно присвоить



любое удобное имя, но лучше руководствоваться общепринятыми правилами именования.

Думаю, что ни для кого не секрет*, что все устройства в Linux представлены в виде особых файлов в каталоге /dev, это справедливо для всех устройств, кроме сетевых адаптеров. Но так было не всегда, в прошлых версиях Linux ядра были доступны устройства /dev/eth0, /dev/tap0 и т.д., в более же новых ядрах эти устройства упразднили, и сетевые интерфейсы были перенесены в память в так называемое пространство сокетов.

Далее я бы хотел рассмотреть особую и очень важную функцию, обеспечивающую обмен управляющими сообщениями между устройством и пользовательским приложениями.

* Комментарий автора.

Следует сказать, что в другой популярной Unix - подобной ОС - FreeBSD по-прежнему, сохранен старый способ. Поэтому, если вы захотите переносить приложения с Linux на FreeBSD - следует учитывать это и другие мелкие различия. Но это не значит, что работа с этими устройствами каким-либо образом осложнилась, все очень и очень просто.

Интерфейсы управления

Для взаимодействия с устройствами, а на самом деле с драйверами устройств, в Unix имеется особый вызов - `ioctl`, означающий Input-Output Control. Справедливости ради, следует отметить, что в Windows имеется подобный интерфейс - `DeviceIoControl`. Для использования данного вызова следует включить заголовочный файл `<sys/ioctl.h>`. Существует также возможность определять свои собственные `ioctl` вызовы, этим пользуются разработчики драйверов. Рассмотрим вызов `ioctl` детально...

```
int ioctl(int d, int request, ...);
```

d - это открытый файловый дескриптор устройства.

request - это тип запроса, для различных устройств запросы различные, соответствующую информацию обычно легко найти в справочных материалов.

третий аргумент - это указатель на `void`, т.е. там могут оказаться какие угодно данные, в зависимости от того, что требует конкретный тип запроса.

В случае успеха вызовом возвращается ноль. В случае ошибки возвращается «-1» и значение глобальной переменной `errno` устанавливается соответствующим образом. Чтобы было понятнее, пример использования (см. листинг 1). Сначала мы подключаем необходимые заголовочные файлы, в которых объявлены используемые нами функции, а так же `ioctl` запросы. В данном примере идет работа с последовательным портом компьютера, а точнее проверяется готовность приема данных. Вызов `ioctl` на открытом файловом дескрипторе, передает драйверу открытого устройства команду `TIOCMGET`, сохраняет и возвращает результат в переменную `Serial`.

Аналогично происходит процесс передачи информации для драйвера из переменной параметра. Как уже говорил, тип переменной параметра зависит от типа запроса. Как правило, это специальная структура. Таков общий принцип

ЛИСТИНГ 1

```
#include <termios.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <errno.h>

int main()
{
    int fd, serial, res; // дескриптор, параметр, результат
    fd = open("/dev/ttyS0", O_RDONLY); // открываем устройство

    if (fd < 0) { // проверяем дескриптор и в случае ошибки и
                // выводим ее пользователю
        printf("Открытие /dev/ttyS0 завершилось с ошибкой: %s\n",
               strerror(errno));
        return 1;
    }

    res = ioctl(fd, TIOCMGET, &serial); // выполняем вызов ioctl
                                        // с запросом TIOCMGET

    if (res < 0) { // проверяем результат и в случае ошибки выводим
        printf("Вызов ioctl завершился с ошибкой: %s\n",
               strerror(errno));
        return 1;
    }

    if (serial & TIOCM_DTR) // проверяем результат
        printf("Последовательный порт не готов\n");
    else printf("Последовательный порт готов\n");
    close(fd); // закрываем дескриптор
    return 0;
}
```

работы `ioctl`, как видим - ничего сложного. Теперь перейдем непосредственно к теме обсуждения - сетевым интерфейсам...

Работа с сетевыми интерфейсами

Выше был рассмотрен пример использования вызова `ioctl` для получения данных из открытого дескриптора последовательного порта. Для сетевых интерфейсов работа выполняется аналогичная. Внимательный читатель мог обратить внимание, как выше я говорил о том, что для сетевых устройств не существует таких специальных файлов, соответственно `eth0` нельзя открыть также, как и последовательный порт. Это так: сетевые устройства перенесены в пространство сокетов и для доступа к этим устройствам следует использовать именно дескрипторы сокетов. Открытие сокета для управления сетевым устройством выполняется так:

```
int sock = socket(AF_INET, SOCK_DGRAM, 0);
```

Результат `sock` и есть дескриптор сокета, перед использованием, его, как и все прочие дескрипторы, следует проверять на отрицательные значения, на случай возможных ошибок. Также, для `ioctl` вызовов на дескрипторе сокета применяется особая структура параметра (`serail`, в примере выше) - `struct ifreq`. Это очень важная структура, используемая во всех случаях работы с сетевыми устройствами. Разберем ее подробнее (см. листинг 2):

```
struct ifreq {
    char ifr_name[IFNAMSIZ];
    union {
        struct sockaddr ifr_addr;
        struct sockaddr ifr_dstaddr;
        struct sockaddr ifr_broadaddr;
        struct sockaddr ifr_netmask;
        struct sockaddr ifr_hwaddr;
        short ifr_flags;
        int ifr_ifindex;
        int ifr_metric;
        int ifr_mtu;
        struct ifmap ifr_map;
        char ifr_slave[IFNAMSIZ];
        char ifr_newname[IFNAMSIZ];
        char * ifr_data;
    };
};
```

ЛИСТИНГ 2

Структура состоит из двух полей: имени интерфейса и объединения, каждое возможное поле, которого выражает конкретный параметр сетевого интерфейса. Данная структура позволяет, как получать параметр интерфейса, так и задавать его. Так как используется объединение – можно получать и задавать только один параметр за раз.

Интересуемые и используемые поля:

```
ifr_addr      - IP адрес интерфейса
ifr_dstaddr   - адрес сервера (для Point-to-Point)
ifr_broadaddr - широковещательный адрес интерфейса
ifr_netmask   - маска подсети
ifr_hwaddr    - mac адрес
ifr_ifindex   - индекс интерфейса (внутри ядра сетевые интерфейсы
                имеют уникальные индексы, для упрощения работы
                сетевой подсистемы)
ifr_flags     - различные флаги (интерфейс поднят или опущен,
```

интерфейс активен или неактивен и др.)

```
ifr_metric    - метрика интерфейса
ifr_mtu       - mtu интерфейса
ifr_map       - структура, содержащая в себе техническую информацию
                (номер прерывания, память устройства)
ifr_slave     - подчиненное устройство
ifr_newname   - новое имя интерфейса (для переименования)
```

Перед любым использованием структуры следует ее обязательно обнулять с помощью `memset`, а затем задавать имя интересующего интерфейса `ifr_name`. Теперь перейдем от теории к действию – получим IP адрес интерфейса `eth0`! Соответствующий пример приведен ниже (см. листинг 3):

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <string.h>
#include <net/if.h>
#include <errno.h>
#include <stdio.h>

int main()
{
    int sock;
    // дескриптор сокета
    struct sockaddr_in* in_addr;
    // структура интернет-адреса (поля)
    struct ifreq ifdata;
    // структура - параметр
    char *ifname = "eth0"; // имя интерфейса

    sock = socket(AF_INET, SOCK_DGRAM, 0);
    // открываем дескриптор сокета

    if (sock < 0) {
        printf("Не удалось открыть сокет, ошибка: %s\n",
               strerror(errno));
        return 1;
    }

    memset(&ifdata, 0, sizeof(ifdata)); // очищаем структуру

    // задаем имя интерфейса
    strncpy(ifdata.ifr_name, ifname, sizeof(ifname));
    // получаем айпи адрес с помощью SIOCGIFADDR,
    // одновременно проверяя результат
    if (ioctl(sock, SIOCGIFADDR, &ifdata) < 0) {
        printf("Не получить IP адрес для %s, ошибка: %s\n", ifname,
               strerror(errno));
        close(sock);
        return 1;
    }
```

ЛИСТИНГ 3


```

in_addr = (struct sockaddr_in *) &ifdata.ifr_addr;
// преобразовываем из массива байт
// в структуру sockaddr_in
printf("Интерфейс %s IP адрес: %s\n",
      ifname,
      inet_ntoa(in_addr->sin_addr));
close(sock);
return 0;
}

```

В этом коде я ввел одну новую структуру и одну новую функцию. Начну со структуры:

```

struct sockaddr_in {
    short          sin_family;
    unsigned short sin_port;
    struct in_addr  sin_addr;
    char           sin_zero[8];
};

```

Даная структура предназначена для хранения базовых данных об адресе сетевого узла. Назначение ее полей:

`sin_family` - семейство адресов, может иметь два значения:

AF_INET для IPv4 и AF_INET6 для IPv6

`sin_port` - порт узла

`sin_addr` - структура адреса (о ней ниже)

`sin_zero` - этот массив можно использовать по своему усмотрению

```

struct in_addr {
    unsigned long s_addr;      // load with inet_pton()
};

```

Эта структура состоит всего из одного поля - числа, представляющего собой собственно IP адрес. Например, «192.168.8.98», в таком формате, имеет вид 1644734656. Функция `inet_ntoa` предназначена для преобразования такого числового значения в привычный цифро-точечный формат. В качестве аргумента, функция принимает `struct in_addr`, а возвращает указатель на строку.

Теперь скажу о преобразовании из массива байт в структуру `sockaddr_in`. Как было показано выше, поле `ifr_addr`, в структуре `ifreq`, имеет тип `struct sockaddr`. Эта структура является своего рода, «упрощенной» структурой `sockaddr_in`:

```

struct sockaddr {
    unsigned short sa_family;
    char          sa_data[14];
};

```

У нее всего два поля: семейство адресов и массив 14 байт, содержащий собственно адрес. Структуры `sockaddr` и `sockaddr_in` хорошо и естественно приводятся к типу друг друга, чем мы и воспользовались. Обратная операция - задание адреса выполняется примерно так же. Отличия только два: перед вызовом `ioctl`, полю `ifr_addr` нужно задать новое значение адреса, а тип запроса будет `SIOCSIFADDR`. Как видно, оба запроса `SIOCSIFADDR` и `SIOCGIFADDR` отличаются на одну букву, которая означает Set и Get, соответственно.

Я не буду приводить пример, показывающий, как задавать новое значение адреса, так как предоставленных сведений уже достаточно для того, что бы читатель разобрался сам. Дам лишь небольшую подсказку: для преобразования строкового значения адреса, например «192.168.8.98», в тип `struct in_addr` следует применять функцию `inet_aton`:

```

inet_aton(const char *saddr,
          struct in_addr *iaddr);

```

`saddr` - указатель на строку с адресом

`iaddr` - указатель на `struct in_addr`

Для получения (или задания) всех остальных параметров используется аналогичный способ, отличие лишь в типе запроса и использовании соответствующего поля в объединении структуры `ifreq`. Список всех возможных типов запроса для получения или задания параметров сетевого интерфейса:

| | |
|---|---|
| <code>SIOCGIFNAME</code> | - получить имя сетевого интерфейса |
| <code>SIOCGIFINDEX</code> | - получить индекс сетевого интерфейса |
| <code>SIOCGIFFLAGS, SIOCSIFFLAGS</code> | - получить/задать флаг интерфейса (о флагах ниже) |
| <code>SIOCGIFMETRIC, SIOCSIFMETRIC</code> | - получить/задать метрику интерфейса |
| <code>SIOCGIFMTU, SIOCSIFMTU</code> | - получить/задать mtu интерфейса |
| <code>SIOCGIFHWADDR, SIOCSIFHWADDR</code> | - получить/задать mac адрес |
| <code>SIOCGIFMAP, SIOCSIFMAP</code> | - получить/задать аппаратные параметры (<code>struct ifmap</code>) |

Наиболее интересные флаги интерфейса:

| | |
|-----------------|--|
| IFF_UP | - интерфейс запущен |
| IFF_BROADCAST | - интерфейс является широковещательным |
| IFF_LOOPBACK | - интерфейс является петлевым |
| IFF_POINTOPOINT | - point-to-point интерфейс |
| IFF_RUNNING | - интерфейс активен |
| IFF_MULTICAST | - интерфейс поддерживает многоадресность |

Небольшой пример использования флагов, получение информации о том, запущен ли интерфейс (см. листинг 4):

```
ioctl(sock, SIOCGIFFLAGS, &ifdata);

if (ifdata.ifr_flags && IFF_UP) {
    printf("Сетевой интерфейс %s запущен\n", ifdata.ifr_name);
} else {
    printf("Сетевой интерфейс %s не запущен\n", ifdata.ifr_name);
}
```

ЛИСТИНГ 4

На данном этапе предлагаю читателю самостоятельно написать небольшое приложение, получающее, в качестве аргумента командной строки, имя интерфейса и выводящее всю информацию о нем. Но как быть, когда заранее неизвестны имена интерфейсов, как получить просто список доступных сетевых интерфейсов? Очень просто. В помощь приходит замечательный вызов `if_nameindex()`.

Небольшой пример, показывающий как получить список всех сетевых интерфейсов и их индексов (см. листинг 5):

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <string.h>
#include <net/if.h>
#include <errno.h>
#include <stdio.h>

int main()
{
    int sock; // дескриптор сокета
    struct sockaddr_in *in_addr; // структура интернет адреса (поля)
    struct ifreq ifdata; // структура - параметр
    // структура интерфейсов и их индексов
    struct if_nameindex* ifNameIndex;

    // открываем дескриптор сокета
    sock = socket(AF_INET, SOCK_DGRAM, 0);
```

ЛИСТИНГ 5

```
if (sock < 0) {
    printf("Не удалось открыть сокет, ошибка: %s\n",
        strerror(errno));
    return 1;
}

ifNameIndex = if_nameindex();
if (ifNameIndex) { // не удалось получить данные
    while (ifNameIndex->if_index) { // пока имеются данные
        memset(&ifdata, 0, sizeof(ifdata)); // очищаем структуру
        strncpy(ifdata.ifr_name, ifNameIndex->if_name, IFNAMSIZ);
        // получаем имя следующего интерфейса

        // получаем IP адрес с помощью SIOCGIFADDR,
        // одновременно проверяя результат
        if (ioctl(sock, SIOCGIFADDR, &ifdata) < 0) {
            printf("Не получить IP адрес для %s, ошибка: %s\n",
                ifdata.ifr_name, strerror(errno));
            close(sock);
            return 1;
        }

        // преобразовываем из массива байт в структуру sockaddr_in
        in_addr = (struct sockaddr_in *) &ifdata.ifr_addr;
        printf("Интерфейс %s индекс %i IP адрес: %s\n",
            ifdata.ifr_name,
            ifNameIndex->if_index,
            inet_ntoa(in_addr->sin_addr));
        ++ifNameIndex; // переходим к следующему интерфейсу
    }
}

close(sock);
return 0;
}
```

Данный пример является доработанной версией предыдущего примера, выводя все доступные интерфейсы, их индексы и IP адреса. На рисунке ниже изображен процесс компиляции и запуска приложения с выводом всей информации. Видно, так же, сообщение об ошибке, для виртуального интерфейса, с которого не удалось получить информацию (см. рисунок):

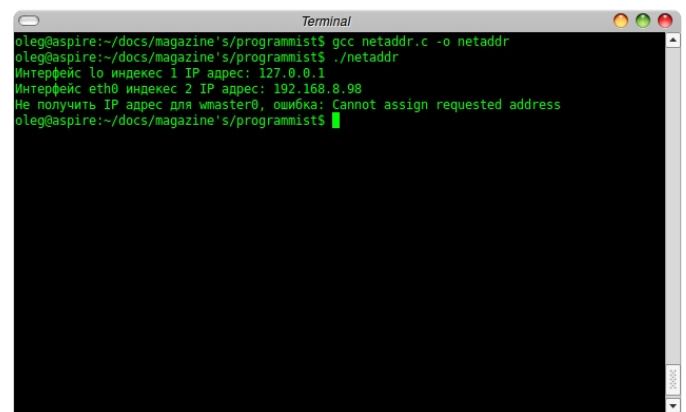


Рис. Процесс компиляции и запуска приложения

Заключение

В данной статье я постарался раскрыть основные аспекты взаимодействия с сетевыми интерфейсами в ОС Linux, многое из этого будет справедливо и для других Unix подобных операционных систем. Надеюсь, что статья получилась интересной и полезной.

Напоследок не могу не сказать, что в Linux, работа с сетью через `ioctl` является устаревшим способом, на смену которого приходит Netlink. Netlink – это модуль, состоящий из двух частей, одна находится в ядре и взаимодействует непосредственно с соответствующими подсистемами, вторая – библиотека на уровне пользователя. Обе части могут обмениваться информацией между собой. Это очень мощный и удобный способ управления всеми параметрами сетевой подсистемы Linux, такие издания, как «Linux journal» рекомендуют пользоваться только Netlink. К сожалению, документации по данному API не так много, как хотелось, и приходится собирать по крупицам. Но в будущих статьях я постараюсь рассмотреть некоторые аспекты использования Netlink, так как уже имею опыт данной сфере. До встречи на страницах журнала!



Литература

- . Исходные коды пакета утилит NET-TOOLS
- . Doxygen документация
- . Christian Benvenuti. Understanding Linux network internals. – O'reilly Media, 2005
<http://www.linbai.info/computers-it/understanding-linux-network-internals.html>
- . Интернет ресурсы, форумы:
www.linuxjournal.com, www.stackoverflow.com

Многие начинающие игроделы сталкиваются с проблемой автоматической прокладки маршрутов ботами на карте. Основных проблем две – генерация вейпоинтов и прокладка по ним кратчайшего (либо оптимального по другим параметрам) маршрута. Данная статья, делает небольшой экскурс по реализации алгоритма поиска кратчайшего пути по ранее установленным вейпоинтам (на основе алгоритма Дейкстры)...



by Utkin www.programmersforum.ru

*На пути постижения мудрости не надо
бояться, что свернешь не туда.*

/ Пауло Коэльо

Прокладка маршрута называется **навигацией**. Она бывает двух видов: автономная – когда объект (бот) самостоятельно прокладывает маршрут из одной точки карты в другую, запоминая его индивидуально (либо в общем, хранилище маршрутов для одной группы юнитов), и предварительная – когда маршруты уже проложены во время проектирования карты (опять же автоматически или программистом), или же на этапе загрузки карты игрового мира. Маршрут обычно представляет собой некоторую совокупность точек (или их координат) со связями между ними, то есть это маршрут, проложенный на **графе**. Сами точки называются **вейпоинтами** – это углы (или вершины) графа. Соответственно подавляющее большинство алгоритмов поиска пути есть алгоритмы по работе с графами.

Краткий экскурс...

Наиболее простой способ – это проложить ключевые точки уже на карте (в момент ее проектирования), а уже на основании имеющейся информации вырабатывать маршрут в зависимости от игрового процесса. Как уже было указано выше, маршрут имеет не только точки, но также и взаимосвязи между ними. В самом простом случае это ссылки на те точки, на которые можно попасть из данной точки, а также отношения между ними (например, это может быть время прохождения или расстояние между точками). В случае если вейпоинты генерируются до игры (во время разработки карты, а не ботом во время игрового процесса) отношения также должны быть уже рассчитаны (например, как расстояния между

доступными точками). Также иногда некоторые маршруты уже изначально заложены и бот «знает» куда идти, но такой вариант должен комбинироваться с алгоритмами самостоятельной выработки маршрута по ряду причин – это делает игровой процесс более динамичным, карта может изменять свои параметры (например, произошел обрыв моста, тогда связи, отношения между двумя точками разрушаются, и требуется новый путь) и т.д.

Особенностью игрового мира является тот факт, что любая ситуация может быть промоделирована, поэтому для простых карт (имеющих малое количество вейпоинтов) можно просчитать оптимальные маршруты до каждой точки на этапе проектирования карты. Для карт, имеющих большое количество вейпоинтов, можно просчитать оптимальные маршруты только для тактически важных точек (какие это точки определяет разработчик), например, от базы до основных ресурсов или от одной лестницы до другой и т.д., остальные маршруты все равно придется просчитывать во время игры.

Предварительный просчет точек осуществлен, например, в игре StarCraft. Это видно если отправить рабочего добывать ресурсы, он смело перемещается через туман войны и по неисследованной области, способен находить подъемы на другой уровень плоскости и сразу определять местоположение моста.

Алгоритм Дейкстры

Этот алгоритм находит кратчайшее расстояние от одной из вершин графа до всех остальных.

Сначала рассмотрим граф без применения алгоритма Дейкстры, это упростит понимание алгоритма в дальнейшем. Для удобства

восприятия обозначим каждую вершину идентификатором, пусть это будет номер вершины (порядок нумерации на графе значения не имеет). В качестве отношения между точками возьмем расстояние. Получается, точка имеет: идентификатор, список точек, куда можно попасть из данной точки и расстояния до каждой из доступных точек.

Вот образец графа (см. рисунок 1):

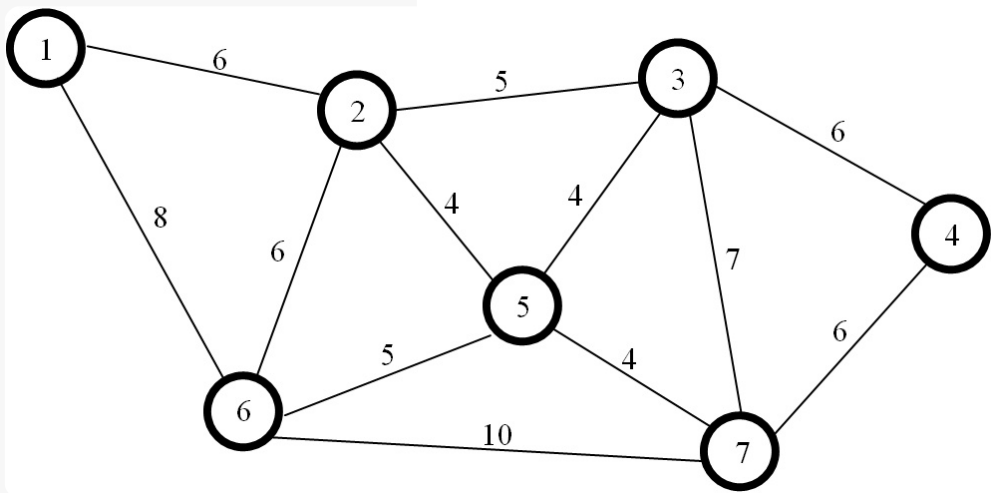


Рис. 1. Образец типичного графа

Представьте, что нам нужно попасть из точки 1 в точку 5. Сколько имеется путей достижения точки 5? Давайте посчитаем:

- а) 1-2-5
- б) 1-2-6-5
- в) 1-6-5
- г) 1-6-2-5

Итого четыре маршрута. Вычислим расстояния по каждому маршруту (суммированием расстояний между точками). Вот расстояния между точками:

- а) 1-2 расстояние 6
- б) 2-5 расстояние 4
- в) 2-6 расстояние 6
- г) 6-5 расстояние 5
- д) 1-6 расстояние 8
- е) 6-2 расстояние 6 (он же 2-6)

Общее расстояние всего маршрута:

- а) 1-2-5 расстояние 6+4=10
- б) 1-2-6-5 расстояние 6+6+5=17
- в) 1-6-5 расстояние 8+5=13

г) 1-6-2-5 расстояние 8+6+4=18

Самый оптимальный (для нашего примера) является путь (а) расстояние всего 10. Если же будут обнаружено два и более маршрутов, имеющих одинаковую длину, то выбирается, как правило, первый (либо заранее продумано, какой из маршрутов, лишь бы выбор был осуществлен).

Также нужно обратить внимание, что на первый взгляд маршруты (б) и (г) равны (казалось бы, от перемены мест слагаемых сумма не меняется), однако следует не забывать, что идентификаторы здесь не несут математического смысла это просто **имена точек**. Таким же образом мы могли бы назвать и (а), (б), (с) и т.д. Операции идут над расстояниями, а суммы пар расстояний (1-2; 2-6) и (1-6; 2-6) не эквивалентны (не равны) между собой.

Теперь сам алгоритм. Он предназначен для поиска всех наикратчайших путей от указанной вершины до всех остальных. Изначально расстояния нам не известны, поэтому будем считать, что они равны максимально возможному расстоянию до каждой из вершин (точек) графа (кроме исходной, до нее расстояние естественно равно нулю). Далее, необходимо отмечать рассмотренные точки графа (чтобы не повторяться)...

Перейдем к алгоритму

Итак, рассмотрим действие алгоритма для первой вершины нашего графа. Вершина 1 имеет отношения (в дальнейшем будем считать отношения просто расстояниями между вершинами графа) с вершинами 2 и 6. Ближайшей точкой будет являться точка 2, поскольку расстояние до нее меньше и составляет 6 единиц (в примере неважно каких единиц, в игре это

могут быть условные единицы реальных км, м и т.д., единицы местоположения юнита на карте, число пикселей с привязкой к координатной сетке области отображения и т.д.). Считаем точку 1 пройденной, поскольку нам известны кратчайшие расстояния до точек 2 и 6.

Следующей рассмотрим точку 2 (потому что она ближе к 1 точке). Для нашего графа соседями точки 2 являются 1, 6, 5 и 3. Точку 1 мы рассматривать не будем, поскольку уже было отмечено, что она была просмотрена ранее. Вот расстояния:

- . до точки 6 расстояние 6;
- . до точки 5 расстояние 4;
- . до точки 3 расстояние 5.

Отсюда следует, что ближайшей точкой к вершине 2 будет точка 5, поскольку расстояние до нее минимально, по сравнению с вершинами 6 и 3.

На данном этапе:

- . расстояние до точки 1 составляет 0;
- . расстояние до точки 2 составляет 6;
- . расстояние до точки 5 составляет 10 (6+4);
- . маршрут до точки 5 следующий – 1-2-5 (и никакой другой до данной точки более не рассматривается);
- . следующей точкой будет являться точка 5;
- . точка 2, также как и точка 1 считается отмеченной и больше не рассматривается;
- . расстояние до точки 6 (маршрут 1-6) равен 8 (0+8), а не 12 (маршрут 1-2-6, расстояние 6+6), поскольку хоть точка 6 и не отмечена, но текущее расстояние до нее уже было вычислено и оно менее текущего (не забываем, что изначально расстояние до каждой точки равно максимально возможному для данного графа).

Собственно этих данных достаточно для того, чтобы выполнить следующие итерации для всех оставшихся точек нашего графа. Несмотря на такое простое описание реализации алгоритма Дейкстры может оставаться трудной, если не представлять, в виде каких структур выражать работу алгоритма. Рассмотрим классический

вариант:

Представление точек графа

Граф можно представлять многими способами, например, в виде таблицы смежности (ru.wikipedia.org/wiki/Список_ребер). Такой способ представления удобен с математической точки зрения, но абсолютно не удобен с практической. Например, точку можно представить как ее координаты и набор отношений (где отношение можно представить как пару – указатель на точку-соседа и расстояние до точки-соседа). Если отношение представляет собой расстояние между двумя точками в данной системе координат, то его можно вычислять автоматически по формуле вычисления расстояния между двумя точками (школьный курс геометрии), при этом расстояние для системы координат с числом осей более двух вычисляется аналогично по обобщенной формуле. Собственно совокупность точек графа и будут являться самим графом.

Отмеченные точки графа

Здесь необходимо отмечать те точки графа, которые были пройдены во время работы алгоритма, чтобы не проходить их повторно. Считаю, эти данные должны лежать отдельно от точек графа, потому как такая информация собственно к графу отношения не имеет (а вот к алгоритму самое прямое).

Таблица кратчайших расстояний

Согласно алгоритму изначально расстояние до точек должно представляться максимально возможным расстоянием (за исключением стартовой, до нее расстояние минимально – 0). Затем эти расстояния заполняются в процессе выполнения алгоритма. Также эти расстояния нужны для сравнения имеющихся и предлагаемых расстояний (расстояние до точки 6 для нашего примера).

Маршруты от начальной до остальных точек

Собственно сами маршруты, по которым в дальнейшем будут осуществляться перемещения. Вообще алгоритм Дейкстры не предназначен для

поиска маршрутов (только расстояний), но его работа построена таким образом, что формирование маршрута по данному алгоритму не представляет никаких сложностей (в момент оценки и внесения в таблицу кратчайшего расстояния):

<http://plagiata.net.ru/?p=90>

. Описание алгоритма Дейкстры

<http://algolist.ru/maths/graphs/shortpath/dijkstra.php>

. Математическое и неформальное описание

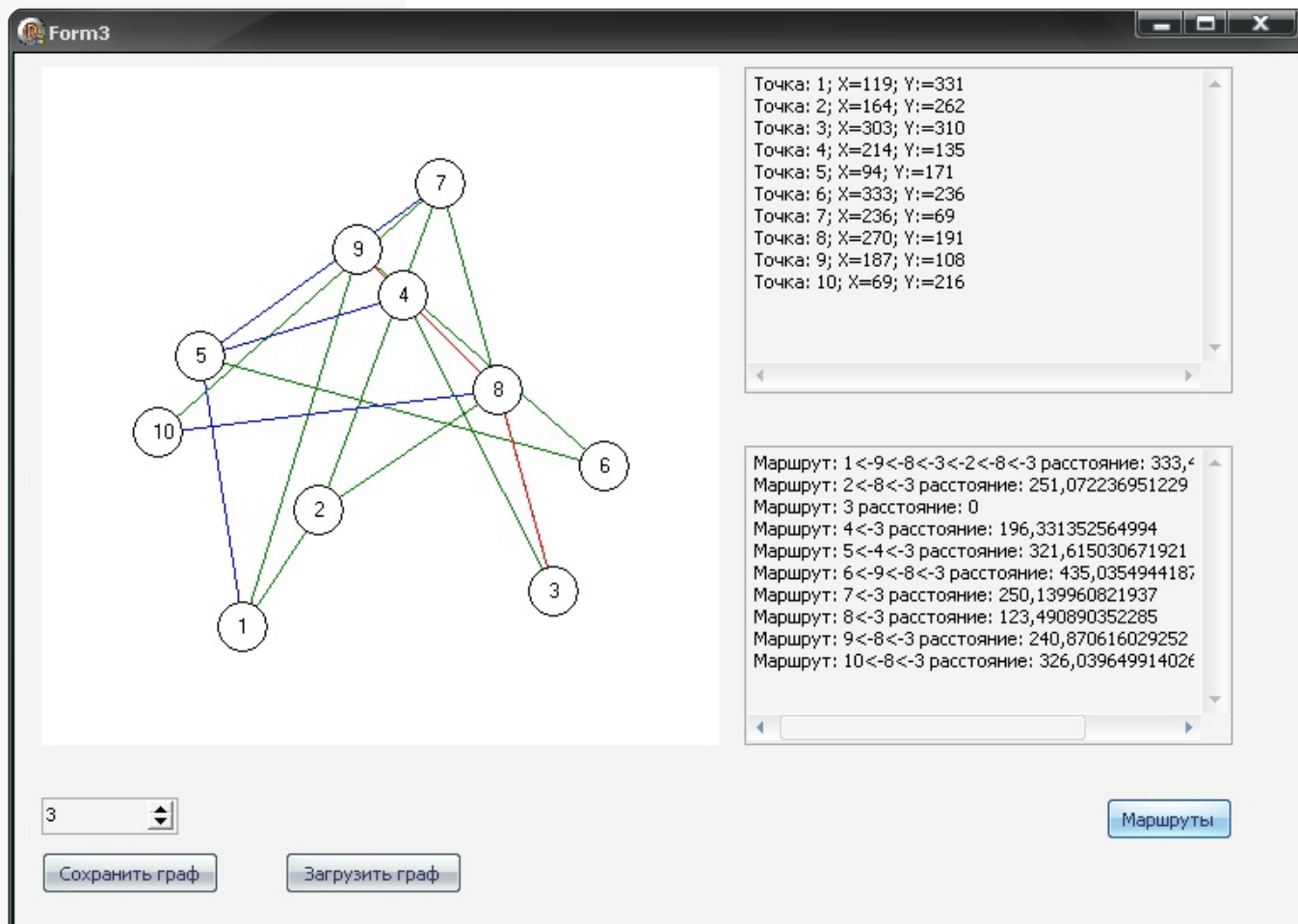


Рис. 2. Тестовая утилита

Заключение

Несмотря на кажущуюся сложность данного алгоритма, при четком понимании работы, его реализация очень проста, а скорость работы вполне приемлема для не очень больших графов.

Исходники тестового проекта прилагаются в виде ресурсов в теме «Журнал клуба программистов. Третий выпуск» или непосредственно в архиве с журналом [5].

Ресурсы

. Реализация алгоритма Дейкстры

алгоритма Дейкстры

http://ru.wikipedia.org/wiki/Алгоритм_Дейкстры

. Дополнительные материалы

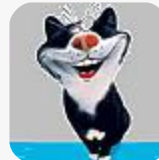
http://borisvolfsen.h11.ru/show_article.php?article_id=0000020 и

<http://www.excode.ru/art6837p1.html>

. Модули и проекты, использованные в статье

<http://programmersclub.ru/pro/pro3.zip>

Здравствуйте, уважаемые читатели. Как и обещал, сегодня с вами мы подробно рассмотрим процедуры работы с графическими объектами, вынесенными в отдельный модуль, позволяющий использовать универсальные методы для создания движущихся изображений, находящихся в файлах, обычно в виде спрайтов...



Продолжение. Начало цикла смотрите в первом и втором выпусках журнала...

Владимир Дегтярь

by DeKot degvv@mail.ru

Создание проекта с несколькими движущимися объектами. Урок 5

Создадим новый проект <Lesson 3> аналогично предыдущим. Введем в него новые движущиеся графические объекты (в папке <data> добавлено еще четыре звездолета. Размер каждого спрайта 100x80 pix. Новые звездолеты будут появляться по случайному закону (используем функцию Randomize) и двигаться будут сверху вниз. Вывод фона и основного звездолета 'ship1' осуществляется также как и в предыдущем проекте <Lesson 2>. Для новых объектов вводим дополнительно BufShipR и BufPicR. Также объявим новые переменные - координаты вывода новых звездолетов и приращения этих координат. Одновременно у нас будут отображаться основной звездолет 'ship 1' и два из 'ship 2' - 'ship 5', выбираемые по случайному закону (см. листинг 1):

```
var
Form1: TForm1;
BufFon, BufFonDop, Buffer: TBitmap;
BufShip1, BufShipR: TBitmap; // буферы спрайтов
BufPicS1, BufPicR: TBitmap; // буферы изображений спрайта

// координаты вывода общего буфера на форму
xf, yf: integer;
// приращение изменения координаты yf по вертикали
dyf: integer;
// координаты звездолета 'ship1'
xs1, ys1: integer;
// приращение координат 'ship1' по гориз. и вертик.
dxS1, dyS1: integer;
// координаты звездолетов 'ship2 - ship5'
xR1, yR1, xR2, yR2: integer;
// приращение координат 'ship2 - 5'
dyR1, dxR2, dyR2: integer;
// номер спрайта и выбор ship2 - ship5
ns, nr1: byte;
nr2: byte = 3;
implementation
```

ЛИСТИНГ 1

В процедуре OnCreate() формы проведем инициализацию буферов и введем начальные данные для переменных. Процедура DrawShipR() для вывода новых объектов ('ship2' - 'ship5') имеет два параметра: i (изменение номера рисунка в файле спрайтов) и j (переменная для номера файла спрайтов). Так как выбор файла спрайта происходит по передаваемому параметру j, то инициализация буфера BufShipR и загрузка в него файла спрайтов находится в процедуре DrawShipR() (см. листинг 2):

```
procedure DrawShipR(i, j: byte);
begin
BufShipR := TBitmap.Create;
// загрузка одного спрайта в буфер рисунка из файла
BufShipR.LoadFromFile('data/ship' + IntToStr(j+2) + '.bmp');
BufPicR.Canvas.CopyRect(bounds(0, 0,
BufPicR.Width, BufPicR.Height),
BufShipR.Canvas, bounds(i *
100, 0, BufPicR.Width, BufPicR.Height));
// зададим прозрачность фона рисунка спрайта
BufPicR.Transparent := true;
BufPicR.TransparentColor := BufPicR.Canvas.Pixels[1, 1];
BufShipR.Free
end;
```

ЛИСТИНГ 2

Все движения организованы в обработчике таймера. Звездолеты 'ship2' — 'ship5' выводятся в Buffer в координаты xR1, yR1 и xR2, yR2 вне видимого окна формы выше. В каждом такте таймера происходит приращение координат для одного dyR1 по вертикали, для другого dxR2 и dyR2 - по горизонтали и вертикали. После того, как объекты выходят за пределы видимого окна формы внизу, вызываются методы random() для задания новых координат xR1, xR2 и номера файла спрайта (nr1, nr2). Координаты yR1, yR2 привязаны к координате yS1, так как 'ship1' неподвижен в координатах окна формы. Функция random(4) возвращает числа в диапазоне 0 .. 3, а файлы спрайтов встречных звездолетов имеют номера 2..5. Поэтому в процедуре загрузки спрайтов BufShipR.LoadFromFile('data/ship' +

IntToStr(j+2) + '.bmp') номер загружаемого файла определяется как IntToStr(j + 2)... В остальном процедуры обработчиков таймера и нажатия клавиш не отличаются от проекта <Lesson 2>.

Использование универсального модуля для работы с графикой. Урок 6

Если рассмотреть внимательно код программы в проекте <Lesson 3>, можно заметить, что многие методы часто повторяются для разных графических объектов (создание буферов, загрузка изображений из файлов, копирование и т.п.). При этом для упрощения, я сознательно применил файлы спрайтов одинакового размера и с равным количеством рисунков в файлах. А если файлов спрайтов будет не пять, а больше и если количество рисунков в каждом файле будет разным? Придется значительно увеличивать код для каждого вида спрайтов. Следовательно, необходимо оптимизировать код программы. Выход здесь в написании методов обработки объектов, не зависящих от количества объектов и применимых для разных изображений.

Данная задача реализована в отдельном модуле <LoadObjectToBufferMod>, позволяющий использовать универсальные методы для создания движущихся графических объектов (находящихся в файлах, обычно в виде спрайтов), имеющих различный размер и разное количество изображений отдельных рисунков.

Модуль находится в папке <Lesson 4> (см. ресурсы к статье). Принцип организации модуля следующий:

- вся работа с графическими объектами проводится через битовые образы TBitmap и области копирования битовых образов TRect
- для работы с фоном используются процедуры InitFon (инициализация) и LoadFon (загрузка фона из файлов)
- функция InitSprite предназначена для инициализации и загрузки рисунков спрайтов
- для вывода фона и изображений спрайтов использован общий буфер типа TBitmap

- в процедуре InitBuff происходит инициализация общего буфера, а в процедуре FreeBuff «переустановка», т.е. уничтожение общего буфера и создание снова, но уже без изображений спрайтов
- в процедуре LoadBuff происходит наложение изображений спрайтов на фон

Подробно работа модуля показана ниже...

Применение модуля LoadObjectToBufferMod

1. procedure InitFon(nw, nh: byte; FileName: string)

Создаем дополнительный и основной буферы фона:

```
BufFonD := TBitmap.Create;
BufFon := TBitmap.Create;
```

Далее загружаем рисунок одного из фонов в дополнительный буфер:

```
BufFonD.LoadFromFile(FileName) или
LoadFromResourceName(hinstance.filename);
```

По загруженному рисунку получаем размер одного рисунка фона (см. рисунок 1):

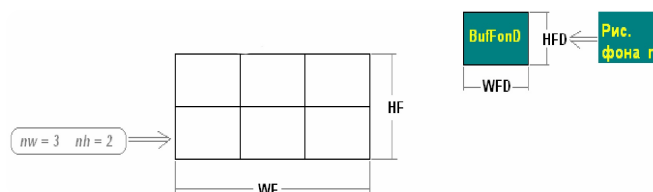


Рис. 1. Определение размера рисунка

Причем, размер буфера фона определяем как:

```
WF := nw * WFD;
HF := nh * HFD;
```

2. procedure LoadFon(xf,yf:integer; FileName: string)

Загружаем все рисунки фонов в буфер фона через дополнительный буфер (см. рисунок 2):

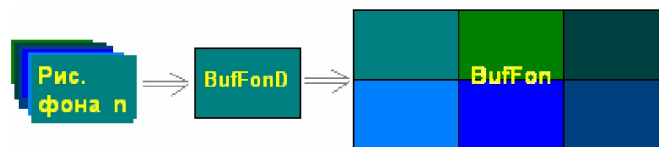


Рис. 2. Загрузка всех рисунков фона

```
BufFonD.LoadFromFile(FileName) или  
LoadFromResourceName(hinstance.filename);
```

3. procedure initBuffer

Создаем основной буфер (Buffer) через который выводим спрайты на форму:

```
Buffer:= TBitmap.Create;
```

Размер основного буфера устанавливаем равным размеру буфера фона WF и HF. Загружаем в основной буфер весь фон (см. рисунок 3):



Рис. 3. Загрузка в буфер фона

```
Buffer.Canvas.Draw(0, 0, BufFon);
```

4. procedure FreeBuffer

Процедура уничтожаем основной буфер Buffer. Применяется когда необходимо убрать какой-либо спрайт с формы:

```
Buffer.Free;
```

Восстанавливаем-же основной буфер с фоном так:

```
InitBuffer;
```

Спрайты, которые должны оставаться на форме, следует перерисовать по новому (см. процедуру InitSprite)...

5. procedure InitSprite(SpriteName: string; N_goriz, N_vertic, N_stroka, N_kadr: byte): byte

Создаем буфер массива спрайтов и загружаем туда файл спрайтов (см. рисунок 4):

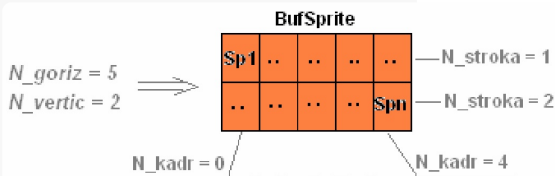


Рис. 4. Загрузка файла спрайтов в буфер спрайтов

```
BufSprite:= TBitmap.Create;  
BufSprite.LoadFromFile(SpriteName) или  
LoadFromResourceName(hinstance.spriteName);
```

Создаем буфер рисунка (одного спрайта):

```
BufPic:= TBitmap.Create;
```

Далее определяем размеры буферов массива спрайтов и буфера рисунка, а также области (типа TRect) загрузки рисунка. Загружаем спрайт в буфер рисунка (см. рисунок 5):

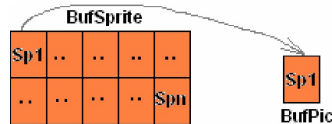


Рис. 5. Загрузка спрайта в буфер рисунка

```
BufPic.Canvas.CopyRect(RectPic.BufSprite.Canvas, rectSprite);
```

Задаем прозрачность рисунку:

```
BufPic.Transparent:= true;
```

Для вывода следующего спрайта функция возвращает (правильнее сказать - функция принимает значение = Result) значение следующего номера спрайта N_kadr:

```
Result:= N_kadr;
```

Уничтожаем буфер массива спрайтов

```
BufSprite.Free;
```

6. procedure LoadBuffer(xf, yf, xs, ys, bs: integer)

В этой процедуре на Buffer выводится участок фона с координатами ранее выведенного спрайта, а затем очередное положение спрайта. Определяем область фона и область дополнительного буфера (см. рисунок 6):

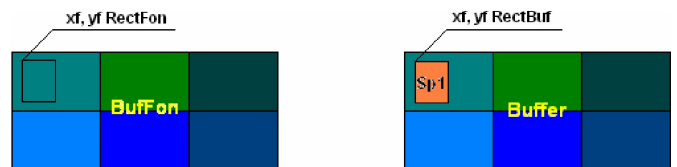


Рис. 6. Определение области фона и буфера

Выводим участок фона в буфер, т.е. затираем спрайт фоном (см. рисунок 7):

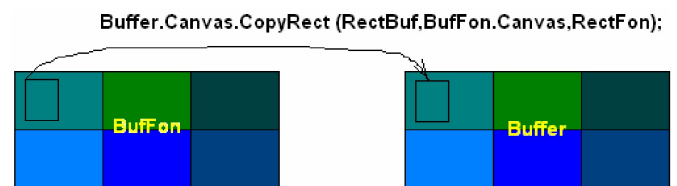


Рис. 7. Затираем спрайт фоном

Выводим очередной спрайт в дополнительный буфер Buffer (см. рисунок 8):

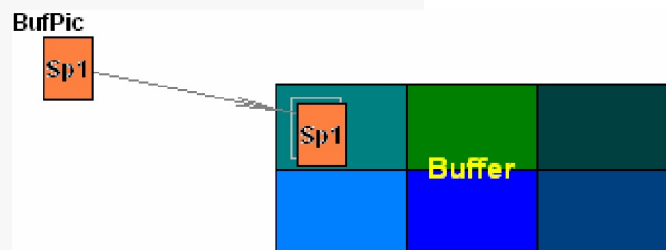


Рис. 8. Вывод следующего спрайта

```
Buffer.Canvas.StretchDraw(Bounds(xs, ys, WP + bs, HP + bs),
    BufPic);
```

Уничтожаем буфер рисунка:

```
BufPic.Free;
```

Далее в программе выводим дополнительный буфер Buffer на форму методом Draw:

```
Form1.Canvas.Draw(x, y, Buffer);
```

Битовые образы фона (BufFon и BufFonD инициализируются (создаются - Create) в программе проекта всего один раз при инициализации программы (обычно вызовом процедуры InitFon в событиях OnCreate или OnActivate). Методы InitBuff, FreeBuff, InitSprite, LoadBuf в программе вызываются неоднократно*. Соответственно и объекты Buffer, BufSprite, BufPic создаются многократно. Поэтому после окончания действия каждого из методов происходит уничтожение битовых образов Buffer, BufSprite, BufPic методом Free.

Комментарий автора.

Загрузку фона можно производить из n - количества файлов, с одинаковым размером не более 1024 x 1024. Для этого вызывать процедуру LoadFon n -раз для разных файлов FonName и изменяя координаты xf и yf.

Модуль можно применять и для простых объектов (один рисунок в файле SpriteName). Присвойте переменным N_goriz и N_vertic значения = 1. Если размер спрайта не изменяется, присвойте переменной bs значение = 0.

Можно загружать рисунки из файлов .jpg. Для этого вместо TBitMap применять класс TjpegImage и в разделах uses LoadObjectToBufferMod и uses Unit1 добавить модуль Jpeg.

Важное замечание.

Перед запуском в среде Дельфи скопируйте в папку с проектом папку data с графическими файлами.

Как работать с модулем?

Для этого необходимо выполнить следующие действия:

1. В процедуре FormActivate (можно в FormCreate) инициализируем буфер фона. Вызываем procedure InitFon(nw,nh: byte; FonName: string) с одним из файлов фонов:

. n раз вызываем procedure LoadFon(xf,yf: integer; FonName: string), последовательно прикрепляя рисунки фонов как бы друг к другу

. инициализируем дополнительный буфер Buffer, вызвав procedure InitBuff. Он получит размер равный сумме размеров всех файлов фонов.

2. Для вывода необходимых спрайтов в нужном месте программы вызываем function InitSprite(SpriteName: string; N_goriz,N_vertic,N_stroka, N_kadr: byte). Функция возвращает очередной номер спрайта для последующего вывода очередного спрайта. Этот номер (N_kadr) необходимо передавать в функцию при каждом ее вызове. Причем, функцию можно использовать для вывода нескольких спрайтов, не забывая передавать ей значение N_kadr для каждых спрайтов.

Далее, вызвав процедуру LoadBuff(), выводим спрайт на канву дополнительного буфера поверх фона. Окончательный вывод дополнительного буфера на канву формы производим методом Draw().

Заключение

Рассматриваемые в данной статье проекты полностью приведены в виде ресурсов в теме «Журнал клуба программистов.

Третий выпуск» или непосредственно в архиве с журналом (папка Lesson3). Продолжение наших уроков смотрите в следующем выпуске журнала «ПРОграммист»...



Многие начинающие программисты пробуют свои силы в создании аудиоплеера. И вот когда у них более-менее получается создать функциональную основную часть и музыка уже играет, хочется добавить еще чего-нибудь крутого. Часто это спектроанализатор или спектрограмма, которую многие ошибочно называют эквалайзером (эквалайзер – это средство настройки, а не анализа звука) [1]. В этом уроке мы попробуем создать спектрограмму на Delphi, а прикручивать ее мы будем к движку BASS. Скажу сразу, что существует Delphi оболочка для BASS от нашего корейского коллеги под названием TBassPlayer, в которой есть спектрограмма, но наша задача – научиться самим, а не использовать готовые решения. Однако вам никто не запрещает просмотреть исходный код этого компонента (он свободный), это никогда не повредит. Изучив этот урок, вы научитесь создавать свою собственную, уникальную своим внешним видом спектрограмму. Желательно, чтобы вы имели основные понятия об ООП, так как я не буду подробно останавливаться на этом и больше внимания уделю вопросам вывода графики...



Александр Терлецкий

by mutabor altair.79@mail.ru

Урок разбит на две части. В первой – пишем класс спектрограммы и отлаживаем его, во второй – прикручиваем его к звуковому движку BASS (вы можете использовать и другой движок, FMOD например, но вам придется разобраться самостоятельно как получить уровни частот). Вариант с самостоятельным разбиением спектра с помощью преобразования Фурье, без использования каких либо звуковых библиотек, я не рассматриваю вообще, если кому очень интересно попробовать, ищите информацию в Интернете, он большой и там все есть*.

Часть 1. Создание класса спектрограммы

Листинг программы смотрите в приложении к журналу [2]. Создадим новый модуль для нашего класса, назовем его Spectrum. В нем описываем главный класс спектрограммы TAnalyzer, и вспомогательные TChannel и TChannels для каналов частот. Класс TAnalyzer мы наследуем от TPaintBox и добавляем к нему несколько нужных нам полей, это:

. Timer (нужен для анимации ниспадающих пиков, в принципе можно было бы обойтись и без собственного таймера, но так наш анализатор получает большую независимость от деталей

реализации использующего его приложения)

. Buffer: TBitmap (буфер нужен чтобы не было мерцания при обновлении изображения)

. Channels (массив каналов, кол-во их будет настраиваемое)

Сами каналы у нас будут иметь градиентную заливку, которая задается полями «LowColor: TColor» и «HighColor: TColor».

Количество каналов и их координаты на экране задается в методе TChannels.SetCount. Для обратной связи с хозяином в классе TChannels предусмотрено поле Owner, которое заполняется сразу после его создания в конструкторе класса TSpectrum. Возможно, здесь я отошел от канонов. Обычно поле Owner в VCL используется немного по-другому, но в данном случае мне было удобнее сделать так и вообще я не претендую на академичность.

Подробно остановлюсь на методе Draw. В нем происходит обновление изображения в такой последовательности. Очищается буфер и заливается черным фоном, затем с помощью Win API функции GradientFill() мы заливаем градиентом все прямоугольники каналов. Функция GradientFill() довольно сложная и имеет много параметров, так что перед ее вызовом много строк кода занимает заполнение нужных структур для передачи в нее. Затем мы проверяем, есть ли пики, которые стали ниже уровня соответствующей им частоты с момента последней

* Комментарий редакции.

Далеко ходить не нужно. Достаточно заглянуть на наш форум <http://www.programmersforum.ru/showthread.php?t=83467> или в блог <http://pblog.ru/?p=658>

отрисовки, если таковые есть, обновляем их положение. Те пики, которые наоборот стали выше текущего уровня, мы пока не трогаем, они будут обработаны в таймере, чтобы они плавно падали вниз (но рисуются все они в этой процедуре, в таймере только смена позиций и вызов отрисовки). После этого мы закрасим черным (цветом фона) верхушки тех каналов, уровни которых ниже максимального. Была использована функция из WinAPI – `OffsetRect()` для смещения прямоугольника канала вверх, а затем, после закрашивания, повторный вызов `OffsetRect()`, только теперь вниз, для восстановления прежних координат прямоугольника канала. И в завершении рисуем пики (короткие горизонтальные линии) для всех каналов. После этого у нас имеется обновленное изображение спектрограммы в буфере, и мы выводим его на экран, а точнее на канву нашего объекта.

Чтобы изображение не пропадало с нашего компонента при перерисовке окна, мы обрабатываем унаследованный метод `Paint` и в ней вызываем нашу процедуру обновления `Draw()`. Хотя с тех пор, как я добавил таймер, это уже лишнее, можете убрать обработку `Paint()` если хотите, но если нет таймера, она необходима.

Класс написан, пора его протестировать. Для этого создадим простое приложение. Для чего, поместим на форму `TGroupBox`, чтобы удобнее было размещать наш анализатор, и две кнопки, по нажатию одной – будет задаваться случайный уровень для всех каналов (см. рисунок 1). По нажатию другой – пройдет волна (см. рисунок 2). Для волны нам понадобится таймер и подключение в `Uses` модуля `Math`, так как мы будем использовать синус. По нажатию третьей – все каналы по очереди заполнятся пиковым значением.

Визуально это будет, похоже, как будто кто-то пальцем провел по клавишам пианино (см. рисунок 3).

Смотрите листинг, компилируйте и разбирайтесь. Если все понятно, то переходим к части 2, где будем подключать анализатор к плееру.

Часть 2. Подключение к BASS

Как это осуществить? Здесь я отдельно рассмотрю два случая, подключение к компоненту-оболочке `TBassPlayer` и непосредственно к BASS. Ну что-ж, приступим...

TBassPlayer

Хочу сразу обратить ваше внимание на нюансы с

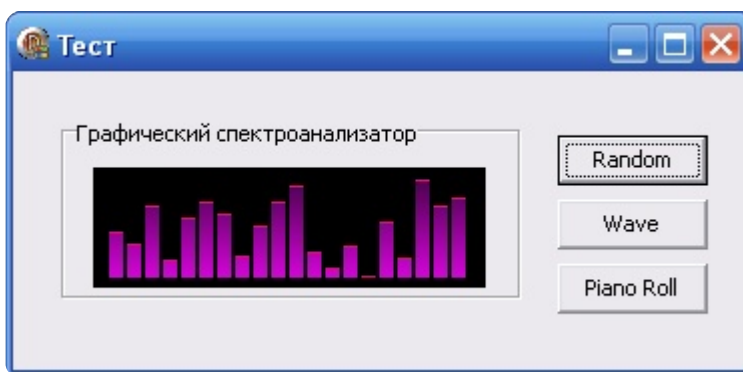


Рис. 1. Тестовая форма.

Спектрограмма случайных величин

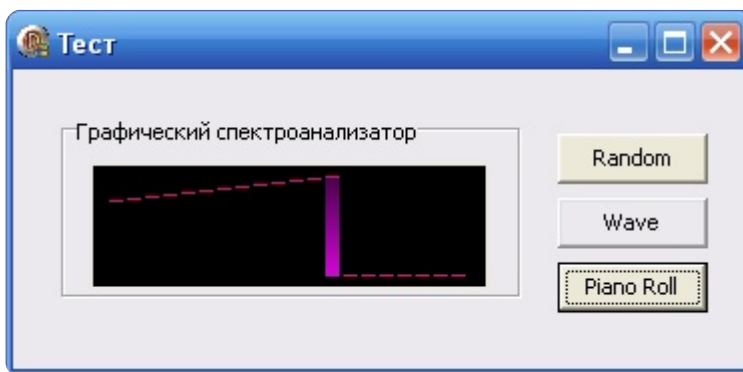


Рис. 2. Тестовая форма.

Спектрограмма волновой функции

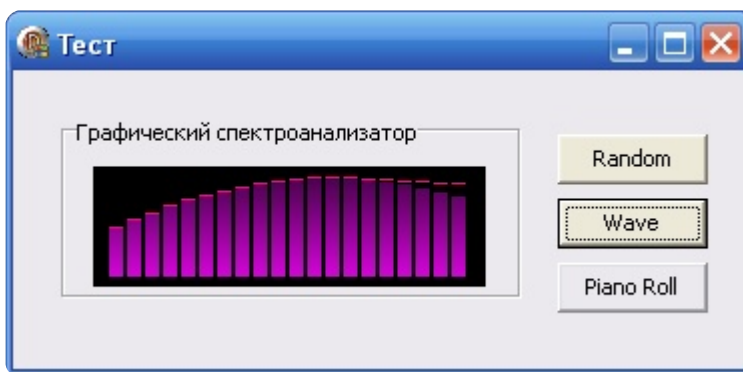


Рис. 3. Тестовая форма.

Спектрограмма в виде пилы

различными версиями как Delphi, так и TBassPlayer. Как вы знаете, начиная с D2009, для строк используется Unicode. Если у вас Delphi более ранних версий, можете дальше не читать и переходить к коду. Если же у вас Unicode версия, то придется внести некоторые изменения в исходники TBassPlayer, в его главный модуль, иначе работать не будет (у меня в D2009 возникали ошибки во время выполнения, связанные с некорректным форматом строк). Хотя в последней его версии и указана совместимость с D2009, да и по времени выхода (май 2009) уже по идее должна быть поддержка юникода, но мне пришлось лезть в исходники даже последней версии. Исправленный модуль вы найдете в файлах к статье [1]. Добавлю еще, что помимо функций оболочки над <bass.dll>, TBassPlayer добавляет поддержку Winamp - плагинов, визуализации и другие функции. Подробнее читайте в справке к компоненту.

Итак, вернемся к нашим баранам. Скопируйте в папку с проектом файл <Spectrum.pas> с нашим анализатором и подключите его в Uses. Я не стал устанавливать в среду TBassPlayer, не люблю я все подряд устанавливать, а просто включил его в Uses. Пути к компоненту можете указать в свойствах среды, а в директорию с программой нужно скопировать файл <bass.dll> подходящей версии (разные версии TBassPlayer работают с разной версией BASS, последняя версия компонента 2.1 может работать с последним BASS 2.4.5). В обработчике события OnCreate() главной формы создаем экземпляр TBassPlayer и назначаем ему на событие OnNewFFTData() процедуру DisplayFFTBand. Затем создаем объект анализатора и назначаем ему количество каналов соответствующее количеству каналов в TBassPlayer (константа NumFFTBands).

Код открытия файла и запуска я скопировал из «демки» к TBassPlayer, заодно там и отображение некоторых свойств файла я оставил, будем выводить их в метки TLabel. Своего я добавил только кнопку Play/Pause, для определения режима работы кнопки используется ее поле Tag

(меняем с «0» на «1» попеременно), при нажатии кнопки читаем ее Tag и производим необходимые действия: меняем название, запускаем или останавливаем воспроизведение и меняем Tag.

Кроме этого, мы создаем процедуру ClearChannels. В ней происходит обнуление каналов, это будет происходить при паузе и остановке. Ну и самое интересное для нас – это метод DisplayFFTBand, который привязан к событию обновления данных спектра, в него приходит состояние каналов в виде объекта Bands: TBandOut. Мы просто перегоняем значения в наш Analyzer.Channels и запускаем обновление Analyzer.Draw. Вот и все. Как видите, подключение элементарное, инкапсуляция проявляет себя во всей красе, все детали скрыты в модулях с классами, и нужно сделать всего несколько телодвижений, для того чтобы приложение заработало.

BASS**

Библиотеку BASS и API для разных языков вы можете скачать с официального сайта проекта www.un4seen.com.

В поставке с BASS идет оболочка для Delphi <bass.pas>. Скопируйте ее в папку с проектом и подключите ее в Uses. Тоже самое сделайте с <Spectrum.pas>. По сути, подключение анализатора мало отличается от случая с TBassPlayer. Также создаем объект, назначаем ему родителя, координаты и количество каналов.

Небольшие отличия есть в механизме получения данных о спектре. Если кто не знает, FFT – означает Fast Fourier Transform, по-русски – Быстрое Преобразование Фурье или БПФ. Таким образом, FFT Data переводится как БПФ данные. В таймере мы получаем от BASS эти самые БПФ данные в массив FData и запускаем процедуру SpectrumRender. В ней проверяется, остановлено

** Комментарий автора.

Пример кода подключения к <bass.dll> и получения данных спектра любезно предоставил наш форумчанин Зарипов Равиль (**ZuBy**). Кстати он разрабатывает плеер на базе BASS, можете подробнее ознакомиться на его сайте www.zubymplayer.com.

ли воспроизведение, если остановлено, то обнуляется массив FData. В связи с этим хочу заметить, что в этом случае наш таймер в классе TAnalyzer является «пятым колесом», так как отрисовка спектра будет вызвана все равно, даже если воспроизведение остановлено. Он не мешает (нагрузку большую он не дает), но и не помогает, если хотите, можете встроить в класс команду по его отключению или вообще убрать таймер, тогда немного переделать функцию Draw нужно будет. Однако, в случае с TBassPlayer собственный таймер нам пригодился.

Но вернемся к процедуре – SpectrumRender. После проверки на остановку, мы заполняем значениями каналы анализатора и вызываем его отрисовку. В отличие от случая с TBassPlayer, где мы просто переносили значения без изменений, здесь мы имеем сырые БПФ данные, и расшифровку их мы должны делать самостоятельно. В данном случае мы берем, из массива с БПФ, данные, начиная с 5-го элемента, в количестве равном количеству каналов в спектрограмме. Правильно это или нет, я не знаю, и похоже мало кто вообще знает как правильно, и в основном все возможно больше ориентируются на красивый вид спектрограммы, чем на точные значения частот. В любом случае, мы каналы не подписываем точным значением частоты, так что с нас взятки гладки.

Автор TBassPlayer применяет другой подход. Можете ознакомиться с ним в исходниках к компоненту [2]. А так как цель статьи – показать, как можно вывести на экран спектрограмму, а не разбор БПФ данных, то на этом скажем Фурье до свидания и продолжим рассмотрение процедуры.

Следовательно, мы уже определились, какие

элементы из массива будем брать. Затем нам нужно умножить значение на коэффициент, найденный пробным путем, он зависит от высоты спектрограммы. Судя по тому, что к полученному значению применена функция модуля Abs(), значения могут иногда быть и отрицательными, оставляем это без изменений (напоминаю, что эта часть кода не моя, я только прикрутил сюда заполнение анализатора). Откидываем дробную часть с помощью функции Trunc(), и можем заполнять этими значениями каналы нашей спектрограммы. После заполнения вызываем отрисовку методом Analyzer.Draw.

Таким образом, при каждом срабатывании таймера, анализатор будет заполняться новыми значениями и будет вызываться его отрисовка. В случае, если трек будет остановлен или поставлен на паузу, так же будет заполняться анализатор, только в этом случае нулями, и так же будет вызываться отрисовка. Это позволит нам увидеть, как пики плавно падают вниз (см. рисунок 4):

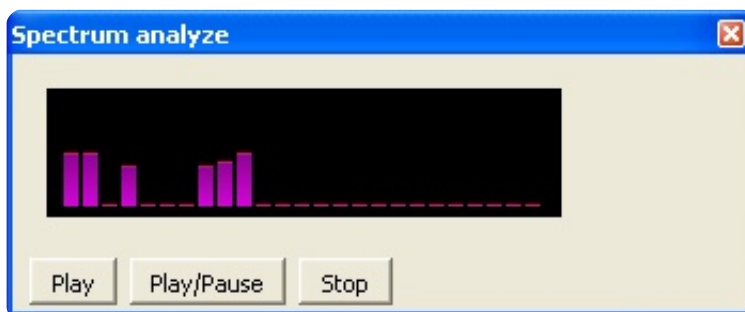


Рис. 4. Отрисовка спектрограммы

На этом закончим рассмотрение подключения к BASS.

Заключение

Итак***, мы создали свой класс спектрограммы, который умеет вывести себя на экран, как по событию перерисовки, так и по таймеру, к тому же он еще и не мерцает при отрисовке и имеет удобный интерфейс. Также мы имеем образец повторно используемого кода, мы ведь его уже использовали как минимум три раза в разных проектах, первый раз в тесте, второй – при подключении к TBassPlayer, и третий раз при подключении напрямую к BASS.

*** Комментарий автора.

Возможно, вы обратили внимание на то, что я не освобождаю глобальные объекты. Так как мы создаем объект в программе только один раз и при его освобождении мы не делаем ничего такого особенного (например, освобождение DLL или других системных ресурсов), то в этом случае я предоставляю освобождение памяти сборщику мусора Delphi, при закрытии программы он вернет ресурсы операционной системе.

Исходники тестовых модулей получения спектрограмм прилагаются в виде ресурсов в теме «Журнал клуба программистов. Третий выпуск» или непосредственно в архиве с журналом.

Ресурсы

- . Типичные примеры обсуждения на форуме по получению спектрограммы через BASS
<http://www.programmersforum.ru/showthread.php?t=7106>
<http://www.programmersforum.ru/showthread.php?t=89708>
<http://www.programmersforum.ru/showthread.php?t=94224>
- . Модули и проекты, использованные в статье
<http://programmersclub.ru/pro/pro3.zip>



Большинство пользователей персонального компьютера привыкли к тому, что весь результат деятельности на компьютере в той или иной степени все равно отражается в самом компьютере. В крайнем случае, отправляется на принтер или в Интернет, или же запись информации происходит на внешние носители (диски, флеш-память и т.п.). И уж мало кто задумывается, что с помощью простого РС – компьютера можно управлять различными внешними физическими устройствами...

Владимир Дегтярь
by DeKot degvv@mail.ru

Тем не менее, на форуме www.programmersforum.ru (да и на других также) часто появляются вопросы: «Как с компьютера зажечь светодиод?», «Как управлять освещением?» или «Можно ли компьютером закрывать шторы на окне?». Ответ: «...однозначно да». С помощью современного персонального компьютера можно создавать, воспроизводить, управлять, хранить, моделировать, обучать и еще реализовать много и много других функций.

Введение

Мы же рассмотрим возможности управления внешними устройствами - при этом возможно управление, как простым вентилятором или светодиодом, так производственными устройствами. Как известно, любой персональный компьютер, да и не только персональный, а и любое компьютерное (еще одно общепринятое название - микропроцессорное) устройство имеет устройства ввода и вывода информации, называемые портами. Для нас пользователей - это просто разъемы для подключения (клавиатура, мышь, модем, флешка и т. д.). Следует заметить, что по одному и тому порту (разъему) можно как выводить информацию, так и вводить. Кроме этого понятие порт - это не только физически внешний элемент (разъем), а больше наоборот - внутренняя логическая структура устройства компьютера, часть его архитектуры.

Думаю, что уже достаточно теории. Перейдем к практической реализации - управлению реальными устройствами с помощью компьютера. Более всего для этого подходит «параллельный»



Рис. 1. «Рабочая лошадка LPT еще даст о себе знать»

порт LPT. Более подробно смотрим о LPT http://ru.wikipedia.org/wiki/IEEE_1284.

Параллельный порт LPT

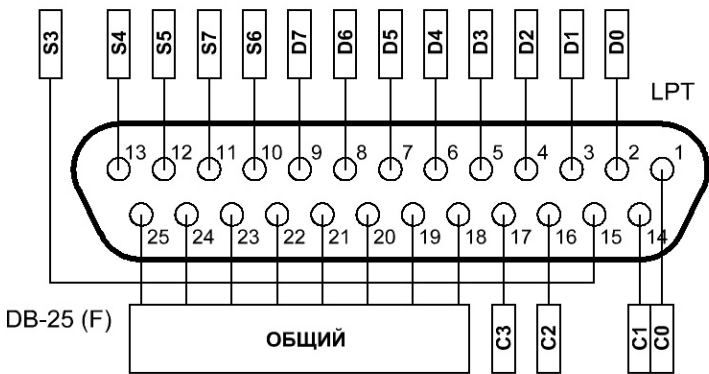
Итак, почему параллельный, а не перпендикулярный? А какие еще бывают? Параллельный, потому что информация через такое устройство передается параллельным способом. А еще может передаваться последовательным - тогда устройство (порт) называется последовательным (последовательные порты компьютера: COM-порт, USB- порт). Наглядно это можно увидеть на рисунке 2:



Рис. 2. Методы вывода информации



Аналогично ввод информации также может быть параллельным или последовательным. Порты (устройства) через которые можно вводить и выводить информацию называют двунаправленными устройствами или устройствами ввода/вывода. Таким и есть параллельный порт LPT, имеющийся в большинстве компьютеров. Посмотрим, что из себя представляет этот порт (см. рисунок 3):



где: C0...C3 – регистры контроля, S3...S7 – регистры статуса, D0...D7 – регистры данных
Рис. 3. Распиновка LPT

Как видите, это такой разъем с 25-ю выводами на задней стенке системного блока компьютера. Итак, имеем физический порт LPT, который фактически состоит из трех регистров. Состав регистров и распределение по контактам разъема приведены в таблице 1:

Табл. 1. Распределение выводов LPT по регистрам «DATA», «STATUS» и «CONTROL»

| \$378 (888) data | | | | | \$379 (889) status | | | | | \$37A (390) control | | | | |
|---------------------|----------------|------------|-------------|---------------------------------|-----------------------|----------------|------------|-------------|-------------|------------------------|----------------|------------|-------------|--------------|
| № контакта разъема | Наименов. цепи | Номер бита | Разрядность | Двунаправленный вход / выход | № контакта разъема | Наименов. цепи | Номер бита | Разрядность | Только вход | № контакта разъема | Наименов. цепи | Номер бита | Разрядность | Только выход |
| 2 | D0 | 0 | 1 | | — | — | 0 | 1 | | 1 | Strobe | 0 | 1 | |
| 3 | D1 | 1 | 2 | | — | — | 1 | 2 | | 14 | AutoLF | 1 | 2 | |
| 4 | D2 | 2 | 4 | | — | PIRQ | 2 | 4 | | 16 | Init | 2 | 4 | |
| 5 | D3 | 3 | 8 | | 15 | Error | 3 | 8 | | 17 | Selln | 3 | 8 | |
| 6 | D4 | 4 | 16 | | 13 | Sel | 4 | 16 | | — | AckInEn | 4 | 16 | |
| 7 | D5 | 5 | 32 | | 12 | PE | 5 | 32 | | — | Direction | 5 | 32 | |
| 8 | D6 | 6 | 64 | | 10 | ASK | 6 | 64 | | — | — | 6 | 64 | |
| 9 | D7 | 7 | 128 | | 11 | Busy | 7 | 128 | | — | — | 7 | 128 | |

Каждый из регистров может содержать байт информации (256 состояний). Часть битов не используется или используется только во внутренней архитектуре компьютера для организации прерываний при работе с принтером или для переключения режимов ввод/вывод регистра «Data».

Регистр данных «Data», номер регистра в шестнадцатеричной системе счисления \$378 (в десятичной 888) - двунаправленный,

восьмибитный. Данные через этот регистр можно как вводить, так и выводить с компьютера, программно устанавливая уровни на выходе порта или же вводить в компьютер, также программно считывая уровни, устанавливаемые внешними устройствами. Регистр управления «Control» \$37A (890). Через него можно только выводить информацию из компьютера. На разъем LPT выводятся четыре младших байта.

Регистр статуса «Status» \$379 (889). Через порт можно только считывать уровни, установленные внешними устройствами. На разъем LPT выведены пять старших байтов.

Таким образом, на разъеме LPT задействовано 17 сигнальных контактов (8 двунаправленных - регистр 888, четыре только на вывод информации - регистр 890 и пять только на ввод информации - регистр 889).

Следует отметить, что в некоторых компьютерах может быть до трех портов LPT: LPT1, LPT2, LPT3 (в современных компьютерах LPT порты зачастую вообще отсутствуют). Адрес самого LPT порта соответствует адресу регистра «Data» и может быть \$278, \$378 или \$3BC. Регистры в соответствующих портах имеют адресацию +1 и +2.

Следующие таблицы (см. таблицы 2, 3) показывают эти особенности данного порта. Для регистра \$378 (888) соблюдается полное соответствие между уровнями напряжения на контактах и логическим кодом. А вот с регистрами \$379 (889) и \$37A (890) все обстоит по-другому:

Табл. 2. Порт \$37A (890) «Control»

| № п. п. | Уровни напряжения на контактах + 5 В – '1', 0 В – '0' | | | | Логический код | | | | | |
|---------|---|----------------------|---|---|----------------|---|---|---|-----|--|
| | | | | | Bit | | | | Dec | |
| | | | | | 3 | 2 | 1 | 0 | | |
| | Разрядность | | | | 8 | 4 | 2 | 1 | | |
| | 17 | Контакты 16 14 | | 1 | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 11 | |
| 2 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 10 | |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 9 | |
| 4 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 8 | |
| 5 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 15 | |
| 6 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 14 | |
| 7 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 13 | |
| 8 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 12 | |
| 9 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 | |
| 10 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | |
| 11 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | |
| 12 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 13 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 7 | |
| 14 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 6 | |
| 15 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 5 | |
| 16 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 4 | |

Управление регистрами

В компьютерах с операционными системами MS-DOS, Windows 9x возможен доступ к портам непосредственно из самой операционной системы, тогда как в системах с NT такой прямой доступ невозможен. Для этих целей используются драйвера в виде библиотек (Inpout32.dll*, WinIO, Giveo). Подключив соответствующие библиотеки к средам программирования, получаем возможность программно работать с портами (считывание

состояния порта или установка выводов порта в необходимое состояние).

Итак, во-первых – помещаем <inpout32.dll> в папке с проектом. Далее в коде модуля Unit после раздела uses размещаем объявления необходимых нам функций из библиотеки:

```
// импорт функций inpout32.dll
uses Windows, Messages, SysUtils, Variants, Classes, Graphics,
    Controls, Forms, Dialogs, StdCtrls, ExtCtrls, ComCtrls;

function Inp32(PortAdr: word): byte; stdcall;
external 'inpout32.dll';

function Out32(PortAdr: word; Data: byte): byte; stdcall;
external 'inpout32.dll';
```

Табл. 3. Порт \$379 (889) «Status»

| № п. п. | Уровни напряжения на контактах + 5 В – '1', 0 В – '0' | | | | | Логический код | | | | | | Dec |
|---------|---|----|----|----|----|----------------|----|----|----|---|-----|-----|
| | | | | | | Bit | | | | | | |
| | Контакты | | | | | 7 | 6 | 5 | 4 | 3 | | |
| | 11 | 10 | 12 | 13 | 15 | 128 | 64 | 32 | 16 | 8 | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 128 | |
| 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 136 | |
| 3 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 144 | |
| 4 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 152 | |
| 5 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 160 | |
| 6 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 168 | |
| 7 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 176 | |
| 8 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 184 | |
| 9 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 192 | |
| 10 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 200 | |
| 11 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 208 | |
| 12 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 216 | |
| 13 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 224 | |
| 14 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 232 | |
| 15 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 240 | |
| 16 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 248 | |
| 17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 18 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 8 | |
| 19 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 16 | |
| 20 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 24 | |
| 21 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 32 | |
| 22 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 40 | |
| 23 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 48 | |
| 24 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 56 | |
| 25 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 64 | |
| 26 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 72 | |
| 27 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 80 | |
| 28 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 88 | |
| 29 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 96 | |
| 30 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 104 | |
| 31 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 112 | |
| 32 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 120 | |

Функция Inp32(PortAdr) возвращает число (тип – байт), соответствующее коду, находящемуся в регистре PortAdr. Функция Out32(PortAdr, Data) возвращает число (Data, тип – байт), которое запишется в регистр PortAdr. Проще сказать Inp32 считывает значение регистра, а Out32

*** Комментарий автора.**
Для работы в среде Дельфи предпочитают библиотеку <inpout32.dll>. Библиотека <inpout32.dll> свободно распространяется в Интернете (см. ресурсы к статье).

устанавливает значение в регистр. Пример применения функций:

```
var val1, val2: byte;
// значение регистра «data» запишется в переменную val1
val1:= Inp32($378);
val2:= 54;
// в регистр «data» запишется число 54 (b 0 0 1 1 0 1 1 0)
Out32($378,val2);
```

При этом следует учитывать, что состояние уровней напряжения на выходных контактах LPT порта соответствует таблице 2 и 3. При необходимости управления отдельными битами регистров можно применить следующие функции преобразований десятичного числа в двоичное и наоборот:

```
// weight (для byte(8 разр.)) = 128, j = 7;
// weight( для word (16 разр.)) = 32 768, j = 15 ;

function Dec_Bin(N_dec: integer; weight: integer; _bit: byte):
byte;
var i , j : byte;
mas_bin: array[0..j] of byte;
N_dec: integer;
begin
for i:= 0 to j do
begin
mas_bin[i]:= N_dec div weight;
if mas_bin[i] = 1 then N_dec:= N_dec - weight;
weight:= weight div 2;
end;

// возвращает значение бита (0 или 1) двоичного числа,
// соответствующего N_dec
Result:= mas_bin[_bit]
end;

function Bin_Dec(weight: integer): integer;
var i , j: byte;
mas_bin: array[0..j] of byte;
N_dec: integer;
begin
N_dec:= 0;
for i:= 0 to j do
begin
N_dec:= N_dec + mas_bin[i] * weight;
weight:= weight div 2;
end;

// возвращает десятичное число, соответствующее двоичному
// в виде массива битов mas_bin [ 0 .. j ]
Result:= N_dec
end;
```

Пример применения функций:

```
var bit : byte;
// переменная bit принимает значение 3-го бита регистра «data»
bit := Dec_Bin(Inp32($378))[3];
```

Вариант тестовой утилиты управления и считывания состояния регистров LPT представлен на рисунке 4 и приведен в ресурсах к статье [1]:

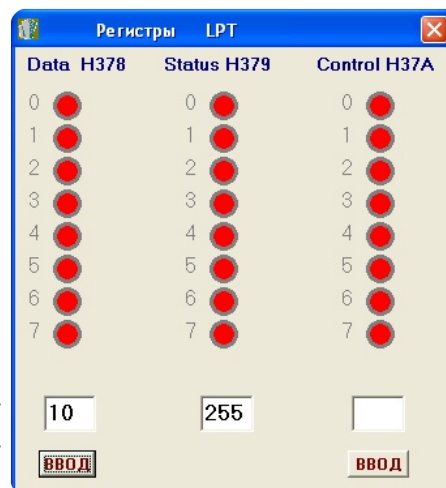


Рис. 4. Утилита считывания и контроля состояний регистров LPT порта

Заключение

В следующей статье мы рассмотрим практические варианты схемотехники подключения порта LPT для ввода и вывода данных. Продолжение смотрите в следующем выпуске журнала «ПРОГРАММИСТ»...

Ресурсы

- Модули и проекты, использованные в статье <http://programmersclub.ru/pro/pro3.zip>
- Сайт Валерия Ковтуна с множеством интересных программ для работы с LPT портом <http://valery-us4leh.narod.ru/main.html>

** Комментарий автора.

Возможность использования регистра «data» (\$378) в качестве порта ввода определяется в настройках BIOS для параллельного порта. Следует установить Parallel Port в EPP. Переключение регистра на «вход» осуществляется программно, путем установки 5-го бита регистра «control» (\$37A) в «1» (при этом все биты регистра «data» устанавливаются в «1»). Следует отметить, что данная процедура возможна не на каждом компьютере и определяется, кроме настроек, еще и конструктивными особенностями порта LPT или материнской платы.

Данная статья будет полезна начинающим программистам, которые никогда не имели дело со звуком и его передачей по сети. Смысл этой статьи заключается в изучении и применении: WINAPI функций ввода и вывода звука WaveIn() и WaveOut() в среде разработки Delphi 7.0, самих компонентов TIdUDPServerSocket и TIdUDPClientSocket. Данные компоненты можно найти в библиотеке Indy, которая в свою очередь находится в свободном распространении на просторах Internet'a...



Александр Владимирович Уколов

by ImmortalAlexSan st_devil@mail.ru

Если вы никогда не программировали в Delphi 7.0, версиями ниже или выше, если вы вообще никогда не программировали на подобных языках, то эта статья не для вас.

Введение

К написанию программы для передачи звука по сети меня побудило желание получить-таки зачет по УИРС (это что-то вроде НИР - научно исследовательской работы студента) у преподавателя, ведущего мой основной предмет, и являющимся моим дипломным руководителем. Перед тем как сесть за Delphi и начать набирать код, предварительно, я изучил кучу литературы в бумажном и электронном виде о принципах упаковки звука и его передачи, о функциях ввода и вывода в самом Delphi и многом другом [1, 2]. Именно ввод и вывод заставил меня задуматься о сложности преподносимого материала. Для человека, никогда не имевшего с этим дело, разобраться в этой области очень сложно, имея под рукой множество кода без комментариев с непонятными процедурами и функциями непонятного WIN API, а если эти процедуры и функции описаны, то это описание предназначено не для начинающих программистов, приходилось все додумывать самому: смотреть подноготную каждой процедуры, и методом проб и ошибок идти медленно, но уверенно к вершине созидания. Но в конечном итоге я добился поставленной цели.

И сейчас, разложив всю информацию, предоставленную мне в кашеобразном виде, по полочкам, я готов поделиться своими знаниями с вами, дорогие читатели! Итак, приступим...

* Комментарий автора.

Описание некоторых вышеуказанных свойств выходит за рамки данной статьи.

Средства разработки

Прежде всего, для работы нам понадобится:

- IDE Delphi версии 7.0 и выше
- Библиотека Indy для Delphi 7.0 (TIdUDPServerSocket и TIdUDPClientSocket) [3, 4]
- колонки и микрофон

Сразу же перейдем к практической части. По мере появления неизвестных функций и процедур в листинге, они будут незамедлительно описываться...

Практическая часть. Создадим клиента

Передача звука в моей программе осуществляется с клиента на сервер, т.е. в одном направлении. Клиент может только писать и передавать, сервер - только принимать и воспроизводить. Первым делом начнем писать клиент. Для этого, создадим новый проект в Дельфи, разместим на форме кнопку TButton и изменим ее свойство Caption на «начать отпавку». После чего, разместим на форме компонент из библиотеки Indy TIdUDPClientSocket (см. рисунок 1):

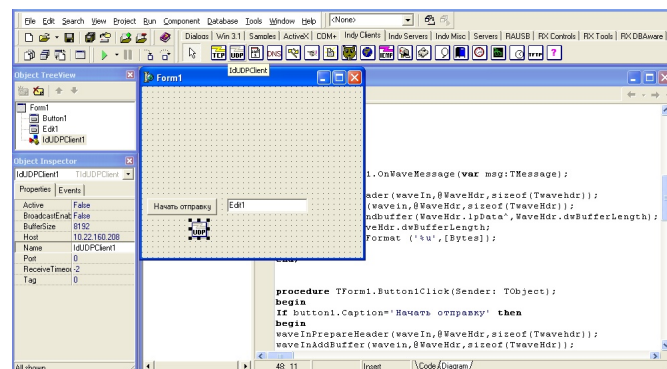


Рис. 1. Режим проектирования формы тестового клиента

Так как тестирование программы будет проводиться на локальном компьютере, то изменим значение свойства «Host» компонента

TidUDPClientSocket на «localhost». Далее я просто перечислю свойства компонента и их значения, что должны быть установлены: Active (false), BroadCastEnabled (false), BufferSize (8192), Name (IdUDPClient1), Port (0), ReceiveTimeOut (-2), Tag (0).

Теперь, нажимаем двойным щелчком по вынесенному на форму компоненту TButton и появится обработчик события Button1Click(), где Button1 – это значение свойства Name данного компонента. В этом обработчике пишем или копируем следующий код (см. листинг 1):

ЛИСТИНГ 1

```

procedure TForm1.Button1Click(Sender: TObject);
begin
    // если на кнопке написано «начать отправку» то:
    // готовим заголовок для буфера, здесь WaveIn – для указания
    // идентификатора устройства ввода (микрофона например),
    // @WaveHdr – указатель на структуру TWaveHdr,
    // sizeof(Twavehdr) – размер данной структуры в байтах

    If button1.Caption='Начать отправку' then Begin
        waveInPrepareHeader(waveIn,@WaveHdr,sizeof(TwaveHdr));
        // заносим данные в буфер
        waveInAddBuffer(wavein,@WaveHdr,sizeof(TwaveHdr));
        // активируем сокет клиента
        IdUDPClient1.Active:= true;
        // считываем данные с микрофона
        waveInStart(waveIn);
        // в едит для наглядности заносим количество записанных байт
        // (делал для себя, чтобы проверять, пишется звук или нет)
        Edit1.Text:= inttostr(WaveHdr.dwBufferLength);
        // меняем название кнопки, чтобы создать возможность
        // прервать отправку пакетов
        button1.Caption:='Остановить отправку'

    end else Begin // если «остановить отправку», то
        button1.Caption:='Начать отправку';
        //закрываем сокет клиента
        IdUDPClient1.Active:=false;
        //разгружаем буфер
        waveInUnprepareHeader(Wavein,@WaveHdr,sizeof(TwaveHdr));
        // приостанавливаем считывание. ЗАМЕЬТЕ! ПРИОСТАНАВЛИВАЕМ!
        // Если мы напишем waveInClose(Wavein), то устройство будет
        // закрыто, и при повторном нажатии на кнопку, не будет
        // никакого результата
        waveInStop(Wavein);
        // смотрим кол-во не записанных байт
        Edit1.Text:=inttostr(Wavehdr.dwBytesRecorded)
    end
end;

```

Вы спросите, а что же такое waveInPrepareHeader? Это функция, выполняющая подготовку буфера для операции загрузки данных. Общий вид:

```

function waveInPrepareHeader(
    hWaveIn: HWAVEIN;
    lpWaveInHdr: PWaveHdr;
    uSize: UINT
): MMRESULT; stdcall;

```

Здесь:

HWaveIn – идентификатор открытого устройства
 LpWaveInHdr – адрес структуры WaveHdr

```

type TWaveHdr = record
    lpData: PChar;           // указатель на буфер
    dwBufferLength: DWORD;   // длина буфера
    dwBytesRecorded: DWORD;  // записанные байты
    dwUser: DWORD;          // пользовательская переменная
    dwFlags: DWORD;         // флаги
    dwLoops: DWORD;         // повтор
    lpNext: PWaveHdr;       // переменная для драйвера
    reserved: DWORD;        // зарезервированно
end;

```

Здесь:

lpData – адрес буфера для загрузки данных
 dwBufferLength – длина буфера в байтах
 dwBytesRecorded – для режима загрузки данных определяет количество
 загруженных в буфер байт
 dwUser – пользовательские данные
 dwFlags – флаги

Могут иметь следующие значения:

WHDR_DONE – устанавливается драйвером при завершении загрузки
 буфера данными
 WHDR_PREPARE – устанавливается системой. Показывает готовность
 буфера к загрузке данных
 WHDR_INQUEUE – устанавливается системой, когда буфер установлен в
 очередь
 dwLoops – используется только при воспроизведении. При записи
 звука всегда 0
 lpNext – зарезервировано
 reserved – зарезервировано
 uSize – размер структуры WaveHdr в байтах

Функция waveInPrepareHeader вызывается только один раз для каждого устанавливаемого в очередь загрузки буфера.

Что такое waveInAddBuffer()? Функция waveInAddBuffer() ставит в очередь на загрузку данными буфер памяти. Когда буфер заполнен, система уведомляет об этом приложение:

```
function waveInAddBuffer(
  hWaveIn: HWAVEIN;
  lpWaveInHdr: PWaveHdr;
  uSize: UINT
): MMRESULT; stdcall;
```

Здесь:

hWaveIn - идентификатор Waveform audio устройства ввода
lpWaveInHdr - адрес структуры TWaveHdr
uSize - размер WaveHdr в байтах

Что такое waveInStart(), waveInStop(), waveInClose()? Общий вид записи таков:

```
function waveInStart(hWaveIn: HWAVEIN):
  MMRESULT; stdcall;
```

waveInStop(), waveInClose() имеют совершенно одинаковый параметр – как и WaveInStart(), которую описывать не имеет смысла, ибо и так понятно, что она начинает считывать данные с устройства ввода, а вот waveInClose() закрывает устройство для записи, и его снова придется открывать с помощью WaveInOpen(), но об этом ниже... А вот waveInStop(), ставит запись как бы на паузу, и нам не надо повторно использовать WaveInOpen().

Что такое waveInUnprepareHeader? Функция аналогичная waveInPrepareHeader(), однако она возвращает выделенную память на буфер, т.е. как бы «уничтожая» его.

Как узнать, что можно передавать данные?

Мы разобрали некоторые функции WIN API, относящиеся к вводу данных. Не устали? Нет? Тогда двигаемся дальше! Создадим собственную процедуру для определения завершения передачи данных в блок памяти посредством WaveInAddBuffer(). А выглядит она так (см. листинг 2):

```
procedure TForm1.OnWaveMessage(var msg:TMessage); ЛИСТИНГ 2
begin
  waveInPrepareHeader(waveIn,@WaveHdr,sizeof(Twavehdr));
  waveInAddBuffer(wavein,@WaveHdr,sizeof(TwaveHdr));
  // отправляем буфер на сервер, где WaveHdr.lpData^ -
```

```
// это ссылка на память, где хранятся считанные с
// микрофона данные уже преобразованные в
// последовательность нулей и единиц,
// WaveHdr.dwBufferLength – длина буфера данных
idUDPClient1.Sendbuffer(WaveHdr.lpData^,
  WaveHdr.dwBufferLength);
// в переменную заносим количество отправленных байт
Bytes:= Bytes+WaveHdr.dwBufferLength;

// формат строки. Посмотрите в google фразу формат дельфи
Caption:= Format ('%u',[Bytes]);
UpDate
end;
```

В этой процедуре используются уже известные вам функции, по этому второй раз описывать их не будем. Пишем её сразу после строки {\$R *.dfm}. А описываем эту процедуру в разделе private класса TForm1, как:

```
procedure OnWaveMessage(var msg:TMessage);
  message MM_WIM_DATA;
```

Эта процедура будет выполняться каждый раз как только передача данных в буфер будет завершена и система сгенерирует сообщение WIM_DATA. Заполним обработчик события формы OnClose() (см. листинг 3):

```
procedure TForm1.FormClose(Sender: TObject; ЛИСТИНГ 3
  var Action: TCloseAction);
begin
  // завершаем все действия
  Action:= caFree;
  // деактивируем сокет
  IdUDPClient1.Active:=false;
  // закрываем устройство записи
  waveInClose(Wavein);
end;
```

И конечно же, заполним обработчик события формы OnCreate() (см. листинг 4):

```
procedure TForm1.FormCreate(Sender: TObject); ЛИСТИНГ 4
begin
  // with – оператор, благодаря которому можно не писать
  // переменные, а указывать сразу их свойства.
  // В данном случае WaveFormat: TWAVEFORMATEX
  // отвечает за сигнал, т.е. за все его характеристики
  with waveformat do begin
    nChannels := 1;
    wFormatTag := WAVE_FORMAT_PCM;
    nSamplesPerSec:= 8000;
    wBitsPerSample:= 8;
```



```

nBlockAlign    := 1;
nAvgBytesPerSec:= 8000;
cbSize         := 0;
end;

// для удобства загоняем размер буфера в переменную,
// которую будем вызывать
bufsize:= waveformat.nAvgBytesPerSec*2 div 16;
// размеру буфера сокета присваиваем размер буфера bufsize
IdUDPClient1.BufferSize:=bufsize;

// waveInOpen опишем чуть ниже, как и обещал,
// WAVE_MAPPER – система сама выбирает устройство
waveInOpen(@Wavein,
           WAVE_MAPPER,
           addr(waveformat),
           self.Handle, 0, CALLBACK_WINDOW);

// выделяем память под заголовок буфера данных
WaveHdr.lpData:=Pchar(GlobalAlloc(GMEM_FIXED, bufsize));
// присваиваем длину буфера TWaveHdr'y
WaveHdr.dwBufferLength:=bufsize;
// сбрасываем флаги
WaveHdr.dwFlags:=0;
// устанавливаем порт подключения для клиента
IdUDPClient1.Port:= 10090
end;

```

Что же такое WaveInOpen()?

Функция waveInOpen() открывает имеющееся устройство ввода Waveform Audio для оцифровки сигнала. Типичная ее структура выглядит следующим образом:

```

function waveInOpen(
  lphWaveIn: PHWAVEIN;
  uDeviceID: UINT;
  lpFormatEx: PWaveFormatEx;
  dwCallback,
  dwInstance,
  dwFlags: DWORD
): MMRESULT; stdcall;

```

Здесь:

lphWaveIn - указатель на идентификатор открытого Waveform audio устройства

uDeviceID - номер открываемого устройства (см. waveInGetNumDevs), это может быть также идентификатор уже открытого ранее устройства. Вы можете использовать значение WAVE_MAPPER для того, чтобы функция автоматически выбрала совместимое с требуемым форматом данных устройство

lpFormatEx - указатель на структуру типа TWaveFormatEx

```

type TWaveFormatEx = packed record
  wFormatTag: Word; // format type
  nChannels: Word; // number of channels (mono, stereo, etc.)
  nSamplesPerSec: DWORD; // sample rate
  nAvgBytesPerSec: DWORD; // for buffer estimation
  nBlockAlign: Word; // block size of data
  wBitsPerSample: Word; // number of bits sample of mono data
  cbSize: Word; // the count in bytes of the size of
end;

```

В этой структуре значения полей следующие:

- wFormatTag** - формат Waveform audio. Мы будем использовать значение WAVE_FORMAT_PCM (это означает импульсно-кодовая модуляция) другие возможные значения смотрите в заголовочном файле MMREG.H
- nChannels** - количество каналов. Обычно 1 (моно) или 2 (стерео)
- nSamplesPerSec** - частота дискретизации. Для формата PCM - в классическом смысле, т.е. количество выборок в секунду. Согласно теореме отсчетов должна вдвое превышать частоту оцифровываемого сигнала. Обычно находится в диапазоне от 8000 до 44100 выборок в секунду
- nAvgBytesPerSec** - средняя скорость передачи данных. Для PCM равна $nSamplesPerSec * nBlockAlign$
- nBlockAlign** - для PCM равен $(nChannels * wBitsPerSample) / 8$
- wBitsPerSample** - количество бит в одной выборке. Для PCM равно 8 или 16
- cbSize** - равно 0. Подробности в Microsoft Multimedia Programmer's Reference
- dwCallback** - адрес callback-функции, идентификатор окна или потока, вызываемого при наступлении события
- dwInstance** - пользовательский параметр в callback-механизме. (сам по себе не используется)
- dwFlags** - флаги для открываемого устройства: CALLBACK_EVENT
- dwCallback** - параметр – код сообщения (an event handle)
- CALLBACK_FUNCTION** - параметр - адрес процедуры-обработчика
- CALLBACK_NULL** - параметр не используется
- CALLBACK_THREAD** - параметр - идентификатор потока команд
- CALLBACK_WINDOW** - параметр - идентификатор окна
- WAVE_FORMAT_DIRECT** - если указан этот флаг, АСМ-драйвер не выполняет преобразование данных
- WAVE_FORMAT_QUERY** - функция запрашивает устройство для определения поддерживает ли оно указанный формат, но не открывает его

Мы использовали callback функцию в

OnWaveMessage(). В последнюю очередь я опишу переменные, которые использовались (см. листинг 5):

ЛИСТИНГ 5

```

type
  TForm1 = class(TForm)
    IdUDPCliet1: TIdUDPCliet;
    Button1: TButton;
    Edit1: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure FormCreate(Sender: TObject);
  private
    procedure OnWaveMessage(var msg:TMessage); message
      MM_WIM_DATA;
    { Private declarations }
  public
    { Public declarations }
    Wavein:HWAVEIN;
    WaveHdr:TWaveHdr;
    bufsize:Cardinal;
  end;

var
  Form1: TForm1;
  WaveDataLength:integer;
  bytes:integer;
  device:word;
  waveformat: TWAVEFORMATEX;
  a:integer;
  
```

Так же для работы программы необходимо добавить модуль MMSystem в раздел uses. Клиент готов! Как видите, не так страшен черт, как его малюют! Перед тем как перейти к написанию сервера, я бы вам настоятельно рекомендовал бы покопаться в генофонде всех выше описанных функций и самостоятельно глубже разобраться в том, как они устроены. Так для более углубленного изучения, советую перевернуть содержимое таких компонентов из серии ACM как Acmln, AcmlOut. Только самообучением можно чего-нибудь добиться.

А что же сервер?

С чистой перед клиентом совестью, можем приступить к написанию сервера! Возможно, эта процедура покажется вам более сложной, но, разобравшись в ней, вы поймете, что это не так. Единственно, что работать мы будем не с одним буфером, а с восемью, для удобства

воспроизведения звука. В один записываем, воспроизводим, очищаем, готовим, записываем и т.д. по очереди каждый из восьми. Так же будет рассмотрена работа с флагами (dwflags) и приема потока данных (TMemoryStream) на сервер. Приступим, нетерпеливые мои!

Как обычно, создадим новый проект и вынесем на форму компонент TMemo (name=memo1) (опять же-таки я использовал его в целях определения получения потока данных, перегоняя его в шестнадцатиричный формат), кнопку TButton и IdUDPServerSocket (см. рисунок 2):

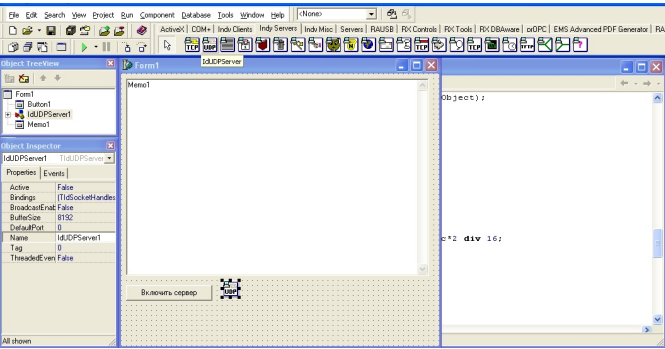


Рис. 2. Режим проектирования формы тестового сервера

Пожалуй, начнем с простого. Напишем ниже приведенный код в обработчике события OnClose() формы (см. листинг 6):

ЛИСТИНГ 6

```

procedure TForm1.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  // завершаем действия
  Action:= caFree;
  // выключаем сервер
  IdUDPServer1.Active:= False
end;
  
```

Далее займемся обработчиком события OnClick() кнопки TButton1 (см. листинг 7):

ЛИСТИНГ 7

```

procedure TForm1.Button1Click(Sender: TObject);
begin
  // активируем сокет сервера
  IdUDPServer1.Active:= not IdUDPServer1.Active;
  if IdUDPServer1.Active then button1.Caption:= 'Выкл сервер'
  else button1.Caption:= 'Вкл сервер'
end;
  
```

Теперь напомним процедуру, которую мы будем использовать для воспроизведения принятого звука (см. листинг 8):

```

procedure TForm1.playound(s:Tstream);
var msg: Tmessage;
begin
    // пока а не равно нашему количеству буферов выполняем
    While a<>CWaveBufferCount do Begin
    // проверку пользовательской установки на то, что буфер
    // готов к записи
    If FHeaders[a].dwUser = 0 then begin
        // записываем в буфер данные из потока, пришедшего от клиента
        s.Read(Fheaders[a].lpdata^,bufsize);
        // процедура waveOutPrepareHeader аналогична процедуре /
        // waveInPrepareHeader
        waveOutPrepareHeader(WaveOut,@FHeaders[a],sizeof(FHeaders));
        // Процедура waveOutWrite аналогична процедуре
        // waveInAddBuffer, только она осуществляет воспроизведение
        // данных из буфера
        waveOutWrite(WaveOut, @FHeaders[a], sizeof(FHeaders));

        mem01.Lines.Add('...Двоичный код потока...');
        // обнуляем флаги буфера/ов в цикле
        FHeaders[a].dwFlags:= 0;

        // уже знакомая нам структура
        With FHeaders[a] do begin
            dwBufferLength := bufsize;
            dwBytesRecorded:= 0;
            dwUser          := 0;
            dwLoops          := 1;
            // А вот здесь мы присваиваем флагу только что
            // воспроизведенного буфера значение, которое отвечает за то
            // что буфер установлен в очередь, т.е. мы как бы
            // циклично используем эти 8 буферов
            dwFlags:= WHDR_INQUEUE
        end;

        // Увеличиваем индекс, чтобы перейти к следующему буферу
        inc(a);
        // соответственно после воспроизведения и подготовки нам
        // больше не нужен цикл и мы выходим из него
        exit
    end
end
end;

```

ЛИСТИНГ 8

Процедура разобрана, осталось ей воспользоваться... Как это осуществить? Все просто, достаточно в обработчике события OnUDPRead() idUDPServerSocket-а написать следующий код (листинг 9):

```

procedure TForm1.IdUDPServer1UDPRead(
    Sender: TObject; AData: TStream; ABinding: TIdSocketHandle);
Begin
    // если мы воспроизвели последний буфер то, начинаем всё
    // сначала (с первого)
    If a = CWaveBufferCount then
        a:= 0;

```

ЛИСТИНГ 9

```

//вызываем нашу процедуру, в скобках пишем наш поток, пришедший
// на сервер, смотрите процедуру сокета
playound(Adata);

// определяем сколько байт мы приняли
Bytes:=Bytes + aData.Size;
// показываем это в названии формы
Caption:= 'Принятых байт' + Format('%u', [Bytes]);
// обновляем форму
Update
end;

```

И не забыть при создании формы проинициализировать наши аудиоустройства. Для этого в обработчике OnCreate() формы запишем (см. листинг 10):

```

procedure TForm1.FormCreate(Sender: TObject);
begin
    bytes:= 0;
    WaveOut:= 0;
    With WaveFormatOut do begin
        nChannels:= 1;
        wFormatTag:= WAVE_FORMAT_PCM;
        nSamplesPerSec:= 8000;
        wBitsPerSample:= 8;
        nBlockAlign:= 1;
        nAvgBytesPerSec:= 8000;
        cbSize:= 0
    end;

    bufsize:= WaveFormatOut.nAvgBytesPerSec*2 div 16;
    For a:= 0 to CWaveBufferCount-1 do
        With FHeaders[a] do begin
            dwFlags:= WHDR_INQUEUE;
            dwBufferLength:= bufsize;
            dwBytesRecorded:= 0;
            dwUser:= 0;
            dwLoops:= 1;
            GetMem(Fheaders[a].lpData, bufsize);
        end;

        IdUDPServer1.BufferSize:= bufsize;
        IdUDPServer1.DefaultPort:= 10090;
        waveOutOpen(@WaveOut,
            WAVE_MAPPER,
            @WaveFormatOut,
            self.Handle, 0, CALLBACK_WINDOW);
    end;

```

ЛИСТИНГ 10

Уважаемые читатели, здесь я пишу без комментариев только для того, что дать вам возможность самим додуматься, что здесь к чему, это не так сложно, тем более, что вы это уже все знаете (мы с вами выше подробно разбирали эти аналогичные функции ввода и вывода и работы с сокетами).

Далее осталось описать переменные и константы (см. листинг 11):

ЛИСТИНГ 11

```
Const
CwaveBufferCount = 8;
type
  TForm1 = class(TForm)
    IdUDPServer1: TIdUDPServer;
    Button1: TButton;
    Memo1: TMemo;
    procedure IdUDPServer1UDPRead(Sender: TObject; AData:
      TStream; ABinding: TIdSocketHandle);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure playsound(s:Tstream);
  private
    hdr: PwaveHdr;
    { Private declarations }
  public
    { Public declarations }
    WaveOut: HWAVEOUT;
    WaveHdrOut, WaveHdrOut2: TWaveHdr;
    WaveFormatOut: tWAVEFORMATEX;
    bufsize: word;
    FBuffer: Pointer;
    FSndBuffer: Pointer;
    FHeaders: array[0..CWaveBufferCount-1] of TWAVEHDR;
    FBufSize: Cardinal;
  end;

var
  Form1: TForm1;
  bytes: Cardinal;

  WaveOut: HWAVEOUT;
  WaveHdrOut, WaveHdrOut2: TWaveHdr;
  WaveFormatOut: tWAVEFORMATEX;
  bufsize: word;
  a: integer;
```

Я не стал описывать процедуру перегонки потока в HEX-формат, так как писал ради передачи данных в ТМемо. В конце концов, вы сами запросто можете убрать ненужные строки, относящиеся к ней.

Заключение

Хочу заметить, что размеры буферов сокетов на сервере и клиенте должны быть равны размерам буферов структуры TWaveHdr, иначе вы не получите никаких звуков на выходе, кроме шипения с прерываниями, равными по длительности размеру вашего воспроизводимого

буфера. Также для более быстрой реакции на события приема звука используйте меньшие размеры буферов, но и соответственно увеличьте их количество (8-ми вполне хватит). При желании, лучше использовать динамический.

Статья была написана специально для форума Клуба ПРОграммистов www.programmersforum.ru. Исходники тестового проекта (клиента и сервера) прилагаются в виде ресурсов в теме «Журнал клуба программистов. Третий выпуск» или непосредственно в архиве с журналом [5].

Выражаю огромную благодарность человеку, чей ник на вышеуказанном форуме **raxp**, который активно помогал мне в изучении этого материала кодами и советами.

Ресурсы

- . П.В. Румянцев. Азбука программирования Win32 API. - М., Радио и связь, 2000, 4-е издание
- . Описание звуковых функций
<http://www.delphikingdom.com/asp/viewitem.asp?catalogid=213>
- . Репозиторий Indy 9 (имя пользователя: Indy -Public-RO) <https://svn.atozed.com:444/svn/Indy9>
- . Репозиторий Indy 10 (имя пользователя: Indy -Public-RO) <https://svn.atozed.com:444/svn/Indy10>
- . Модули и проекты, использованные в статье
<http://programmersclub.ru/pro/pro3.zip>
- . Обсуждение на форуме разработки прототипа VoIP телефона <http://www.programmersforum.ru/showthread.php?t=91506>

Здравствуйте, уважаемые читатели! Для начала оговорим, что мы решили разбить статью на несколько частей, так как данный материал не поместится в объеме одного выпуска. В первой части вы узнаете о написании ленты новостей, гостевой книги и вообще всего интерфейса пользователя, а остальные будут посвящены написанию панели администратора. Сегодня рубрику буду вести я, Егор Горохов aka **Revival001**, далее меня сменит Алексей Шульга aka **Levsha100**. Мы будем помогать друг другу в этом нелегком труде. Итак, начнем...



Егор Горохов

by Revival001 revival1002@gmail.com

достаточно сменить фоновую картинку и немного подкорректировать таблицы стилей CSS.

1. Дизайн

Для начала был нарисован PSD шаблон сайта в графическом редакторе Photoshop. Оттуда же мы потом и «выдирали» картинки для сайта. В

2. Верстка

Для создания закругленных углов использовалось свойство *border-radius*. К сожалению, пока что свойство работает не во всех браузерах, например



Рис. 1. Предварительный редизайн сайта журнала

результате у нас получилось следующее (см. рисунок 1). Плюс такого дизайна в том, что можно очень быстро сделать редизайн. Для этого

в IE 6...8 оно игнорируется. Конечно существуют и альтернативные способы создания круглых углов, но они зачастую требуют до 9 (!) картинок блока

(4 угла, 4 стороны и центр). Так как мы «очень ленивые», да и острые углы не очень портят дизайн, мы использовали простейшее решение, с надеждой на то, что вскоре все браузеры будут поддерживать данное свойство...

Для полупрозрачности главного `div` не было найдено (возможно мы плохо «гуглили») кроссбраузерного решения, поэтому мы пошли на хитрость, создав однопиксельный полупрозрачный `png`, и поставили его на фон. Но и этот способ, как оказалось имеет свои минусы. Например*, ненавистный всеми дизайнерами/верстальщиками IE6 его игнорирует.

* Комментарий автора.

Кстати, я уже давно не видел людей использующих IE6. Все мои знакомые, даже если они мало понимают в компьютерах (например девушки) используют браузеры типа Opera и Mozilla Firefox. Поэтому причин продолжать верстать под MSIE6 я уже не вижу.

Ознакомится с CSS можно по адресу: <http://procoder.info/style.css>

3. Написание скриптов

Для начала, вынесем все наиболее часто используемые переменные в отдельный файл `<config.php>`, и потом будем его «инcluirить» во все скрипты. Мы поместили туда логин/пароль от БД, путь к сайту и т.п. В дальнейшем, если мы захотим изменить что-либо, то это займет всего лишь несколько секунд, и не потребуются вносить изменения во множество скриптов.

Итак, при заходе на сайт пользователь должен видеть: ленту новостей, журналы доступные для скачивания, гостевую книгу. Реализация происходит практически одинаково. Покажу на примере **гостевой книги** (см. листинг 1):

```
<?php
require_once "config.php"; // Подключение файла конфигурации
// Функция обрабатывает строку, и заменяет все bb-коды на html
function bbcodes($str) {
    $bbcode = array(
        "/[b\](.*)\[\/b\]/is" => "<strong>$1</strong>",
        "/[u\](.*)\[\/u\]/is" => "<u>$1</u>",
        "/[url\=(.*)\](.*)\[\/url\]/is" => "<a href='$1'>$2</a>",
        "/[color\=(.*)\](.*)\[\/color\]/is" => "<font color='$1'>$2</font>",
        "/[i\](.*)\[\/i\]/is" => "<i>$1</i>",
        "/[quote\](.*)\[\/quote\]/is" =>
```

ЛИСТИНГ 1

```
"<div class=\"quote\">$1</div>", "/[img\](.*)\[\/img\]/is" =>
"<img src=\"$1\">";
$str = preg_replace(array_keys($bbcode), array_values($bbcode),
    $str);
return $str;
} // Функция взята из мануалов по PHP

// Подключение к БД
$db = mysql_connect($db_server, $db_user, $db_pass);
mysql_select_db($db_name);
// Выбираем из таблицы table_gb последние 30 записей
// и сортируем их по id
$db_query=mysql_query("SELECT * FROM `table_gb` ORDER BY `id`
DESC LIMIT 0 , 30", $db);
while ($res = mysql_fetch_array($db_query)) // Вывод записей
{
    echo "<div class=\"guest_comment\"><b>".$res['name']. "-
    ".$res['date']. "</b><br>";
    echo bbcodes(wordwrap("<pre>".$res['text']. "</pre>", 75));
    echo "</div><br>";
}
?> // Тут находится форма, для добавления записей в БД,
// которая передает скрипту gb.php данные
```

А теперь скрипт добавляющий записи в БД (см. листинг 2):

```
<?php
require_once "config.php";
$db = mysql_connect($db_server, $db_user, $db_pass);
mysql_select_db($db_name);

// Получаем данные и убираем из них спец-символы,
// для защиты от XSS-атак
$name = htmlspecialchars($_POST['name']);
$text = htmlspecialchars($_POST['text']);
$date_array = getdate(time());
$date = $date_array['mday']. "." . $date_array['mon']. "." . $date_array['year'];
if (($text=="") or ($text=="Введите текст"))
{
    echo "<script>alert('Запись не добавлена.');

```

ЛИСТИНГ 2

Аналогичным образом выводятся записи для новостей, журналов для закачки и другая информация (см. рисунок 2):

**** Комментарий автора.**

Мы с Алексеем заметили, что у нас серьезные проблемы с безопасностью, поэтому если вы нашли уязвимость, то пожалуйста сообщите о ней на ящик журнала reddatacentr@gmail.com.



Рис. 2. Скрин тестовой работы скриптов

Подведем предварительные итоги...

Что есть и что планируется? Пока сделано очень мало, и нововведения** будут появляться практически каждый день. Совсем скоро будет система регистрации, что позволит комментировать статьи, новости и выставлять им оценки. У каждого пользователя будет личный кабинет. Создавать отдельный форум для сайта не планируется, так как это совместный проект Клуба ПРОграммистов www.programmersclub.ru и форум уже есть. Существующие алгоритмы тоже будут улучшены. Все предложения оставляйте либо на форуме, либо отправляйте на электронный ящик редакции.

Мы изучили Ваше коммерческое предложение по разработке информационной системы и приняли решение приобрести некоторое количество травы, которую вы курите.

Разговаривают два программиста:

- А у меня микроволновка не включается...
- Что пишет?

Заходит жена программиста в комнату и говорит мужу, который поглощён работой за компьютером:
- Твой сын может читать только печатные буквы, а прописные не понимает!

Программист, не отрываясь от компьютера:

- Поставьте более новую версию «FineReader»... (пауза)
- Что ты сказала, дорогая?

Новый вирус - «Бомж!» Он просто роется в «Корзине»...

В отдельном кабинете за столом сидит админ, читает журнал. Дверь открывается, влетает чуть не плача девушка и запинаясь говорит - «у меня Ворд завис, не знаю что делать...» Админ, добродушный дяденька, не меняя позы и не отрываясь от журнала говорит: «Выйдите и зайдите снова». Девушка, покраснев, выходит из кабинета закрыв дверь. Через три секунды дверь вновь открывается и девушка запинаясь еще больше: «У меня Ворд завис, не знаю что делать...»

Имя: Екатерина
Фамилия: скрыто
Дата рождения: скрыто
E-mail: ivanova1985@mail.ru

Одна девушка-программистка характеризует своего знакомого: «У него в алгоритме ухаживания ни одного if не предусмотрено, не говоря уже о таких сложных конструкциях, как while или until!»

Трудное детство... килобайтные игрушки...

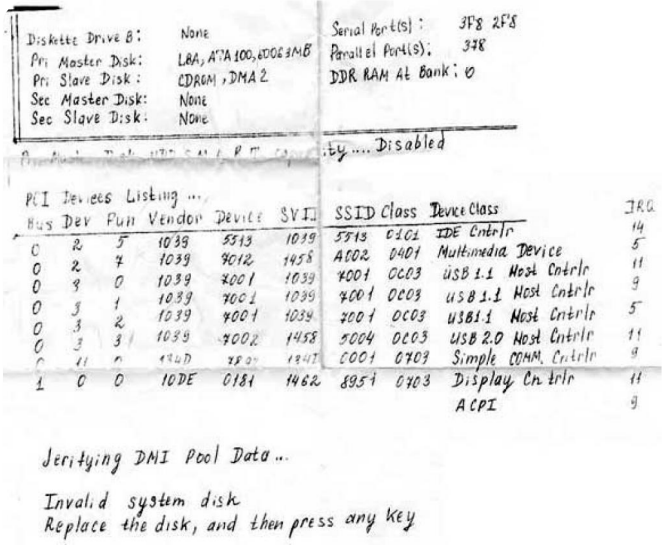
- Для чего нужны бухгалтерские программы?
- Как для чего? Для снижения безработицы,

конечно! Там, где раньше сидел один бухгалтер, теперь сидят еще оператор, программист, электронщик, пять наладчиков и начальник отдела вычислительной техники...



Программист жене: CD молча и не DVD меня до белого каления!!!

Если у вас нет отца, то кликните на рабочем столе правой кнопкой мыши и дальше выберите «Создать / ПАПКУ».



Приходит веб-мастер сдавать на права. Опрос:

- Цвет глаз
- Пишет: #44ff99

Заядлый компьютерщик в ЗАГСе интересуется:

- А Вы регистрируете детей с именами длиннее 8 букв?

Гаишник тормозит машину... Открывается водительская дверь и вываливается пьяный мужик.

- Ваши права!
- Аддмминистратор!

Идет программист по стройке, вдруг с крана срывается огромный блок и падает программисту на голову.

Тетрис, успел подумать программист.

Новая русскоязычная поисковая система «Иван Сусанин».

Кнопка с крестиком в правом верхнем углу экрана – шлюз в реальный мир...

Был случай лет 15 назад... мы тогда прикручивали одну из первых АСУ ТП к ДНС-КНС в Западной Сибири. Написали голосовых сообщений, даже усилитель поставили, чтобы оператор на улице слышал. Все бы ничего, но мне вздумалось повеселиться. Записал такой текст: Внимание, аварийная ситуация! Нарушен технологический режим! Установка будет взорвана через 5 минут. Срочная эвакуация! Ну и тому подобное с обратным отсчетом. Получилось очень правдоподобно. А потом... прикрутил сие произведение на алгоритм, который вообще никогда (как я думал) не произойдет, потому что это прямое нарушение техпроцесса. Однако фигушки! Произошло. Через месяц после

внедрения. Оператор там совсем безбашенный на вахту заступил.

Ну че, я на том объекте больше не появлялся: тамошние горячие башкирские мужики грозились меня прирезать... ибо убегали они в 40- градусный мороз при сработке моего аларма в лес, километра 3 от установки. Причем спросонья, в одних тапочках по сугробам. На разборке у руководства предъявил алгоритм, оператора наказали... но, в общем меня тоже по головке не погладили за шутку. Хотя отнеслись с юмором и пониманием того уровня знания техпроцесса операторами.

Аналогичная установка на Саранском пивном заводе. Там местный умелец прикрутил к WinCC

Источник: "Персональные ЭВМ в инженерной практике". - М.: Радио и связь, 1989г.

"...Одним из примеров громоздкой и, по мнению авторов, бесполезной надстройки является интегрированная система WINDOWS фирмы Microsoft. Эта система занимает почти 1 Мбайт дисковой памяти и рассчитана на преимущественное использование совместно с устройством типа "мышь"... ..Таким образом, читатель уже понял, что среди надстроек над ДОС бывают довольно бесполезные системы, которые только выглядят красиво, а на самом деле отнимают время пользователя, память на дисках и оперативную память ЭВМ. Обманчивая красота таких систем, однако, сильно воздействует на неискушенных пользователей, которые не имели практики работы на машине. Инерция мышления бывает столь сильна, что авторам приходилось наблюдать, как люди, начавшие работать с подобной настройкой, впоследствии с трудом заставляют себя изучать команды ДОС. Хочется предостеречь от этой ошибки читателей..."

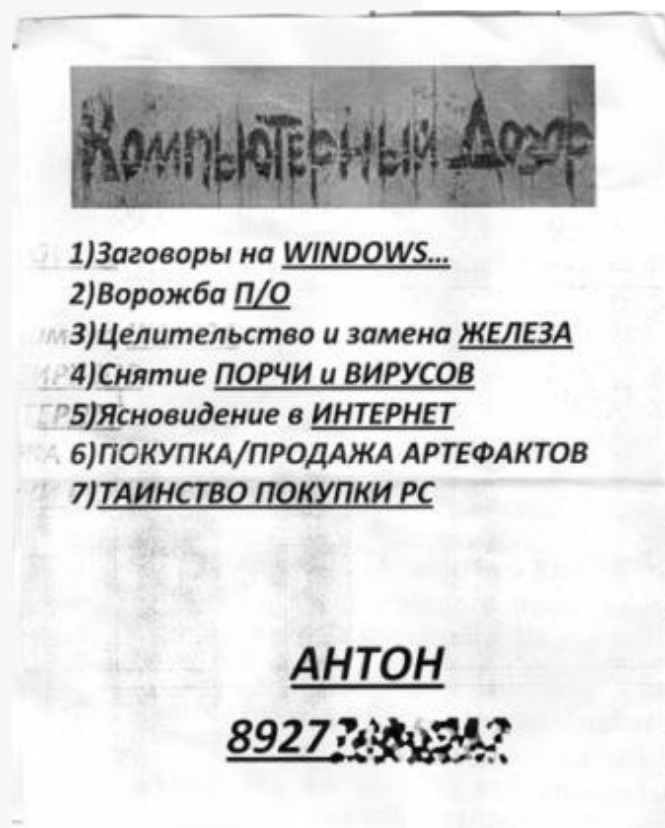
сообщения, зачитанные приятным речевым движком. Только сделал он это как-то странно: записал текст, скормил движку,

получил .wav'ы и их уже прикрутил. Не судьба была сразу прикрутить? Операторы нам жаловались, что по ночам спать мешает. А так как сеть вся MPI (вы никогда не видели 5-7 проходных коннекторов MPI/Profibus, вставленных в одно гнездо друг за другом так, что шкаф не закрывается?) и длина ее критическая, то раз в 10 минут все отваливается и восстанавливается снова. Соответственно женщина из ящика сначала говорит, что все отключилось, а потом, что все включилось. Процесс внушает! Но, больше всего доставал один дедушка - оператор. Он с ней разговаривал:

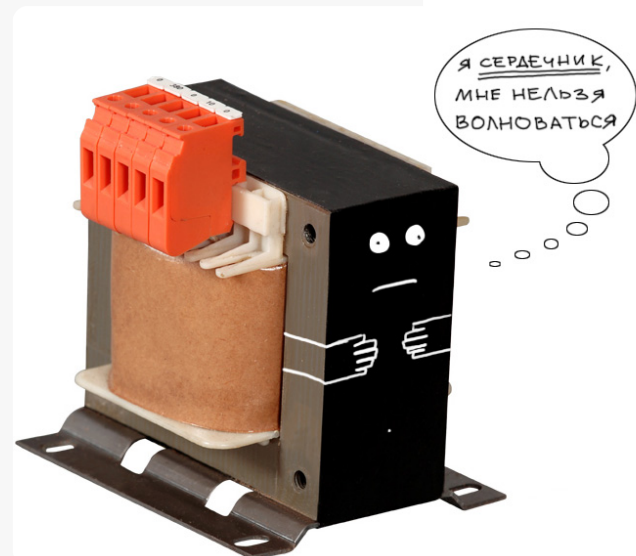
- Афанасий, крепкое запущено на фильтрацию.
- Правильно!
- Сепаратор разогнан.
- Молодец!

Заходит программист к врачу:

- Доктор, я кашляю. Это к вам?
- Да, вы попали по адресу. Я - терапевт.
- Ой, нет. ТЕРАпевт - слишком круто! Мне бы к ГИГАпевту. Или даже к МЕГАпевту!



Задача, которую хороший программист решит в минуту, а хороший физик сойдет с ума: Ася весит 4,2 метра. За сколько времени она скачается, если ширина канала - 5 кило в секунду?



- С вас 283 рубля. Завернуть?
- А? Нет, не надо, просто в пакет положите... Один

вопрос - гарантия есть?

- Простите, а для какого случая гарантия?
- Ну, если не прочитается?
- То есть как?
- Ну несовместимость требований, например... Брак там, туда-сюда...
- Вся прогрессивная молодежь в курсе, что обычно сначала туда-сюда, а потом брак. В худшем случае... Не прочитается... А, я понял - вы по профессии, как бы это сказать, минимум продвинутый пользователь?
- Да, я программист. Но как вы догадались?
- Молодой человек, будь на моем месте Шерлок Холмс, он бы сказал - по очкам, грязным джинсам, торчащему из кармана рюкзака винчестеру, трем бутылкам пива в руках и клавиатуре под мышкой. Но я не он.... И все намного проще. Это бумажная книга, она обязательно прочтется...

- Алло?

- Алло! Доброе утро, молодой человек! Умоляю вас, не кладите трубку! Скажите мне - кто я?! Где я сейчас?! Куда мне идти?!



Дорогие радиослушатели, вы прослушали короткую радиопостановку «Утро DHCP сервера».

Коктейль для программиста:
водка с пивом
640 на 480.

