Внимание: **Эта документация взята с сайта http://www.opennet.ru**

The Linux Cyrillic HOWTO

В этом документе описывается, как настроить Linux для редактирования, просмотра и печати документов на русском языке.

Оглавление.

1. Общие примечания

1.1 Введение

В этом документе описываются приемы которые нужны для редактирования, просмотра, и печати документов с использованием кириллических символов (в основном это относится к русскому языку) под Linux. И хотя здесь предполагается, что вы используете Linux как операционную систему, большинство описываемой информации одинаково применимо к другим разновидностям Unix. Я попытаюсь указывать на различия.

Имеется ряд популярных дистрибутивов Linux. Для описания приемов работы в качестве примера я выбрал дистрибутив RedHat Linux. Однако, вы найдете, что практически вся приведенная здесь информация подходит и для вашего "любимого" дистрибутива.

Предполагается, что любая операционная система UNIX настраивается и поддерживается опытным человеком. Одного знакомства с книгой из серии "что-то там для чайников", как правило, для этого дела недостаточно. Кириллизация операционной системы модифицирует ее, следовательно требует определенных знаний о том что вы делаете. Несмотря на то, что я пробовал упростить изложение настолько, насколько это возможно, наличие некоторого опыта работы с настраиваемым программным обеспечением является преимуществом на пути "всеобщей кириллизации". Я не собираюсь описывать здесь что такое X Windows или как создаются документы в системах TeX и LaTeX, или как установить и настроить принтер в Linux. Это описано в других документах.

UNIX это многопользовательская система и поэтому условия при котором вы проводите кириллизацию могут изменяться: вы можете быть системным администратором (или владельцем системы) пытающимся кириллизировать всю систему. С другой стороны, вы можете быть обычным пользователем, не имеющим привелегий системного администратора (root или superuser) и вы хотите кириллизовать систему только для себя. Большинство описываемых мной программ имеют достаточно гибкую настройку, позволяющую сделать ее и с точки зрения всей системы, и с точки зрения обычного пользователя. Я попытаюсь отметить оба случая.

ВНИМАНИЕ: X Windows, TeX и другие компоненты Linux- сложные системы с "навороченной" настройкой. Если вы сделаете что-либо неправильно, то не только потерпите неудачу с русификацией системы, но и можите частично, если не полностью, испортить систему. Это не должно вас пугать, это просто предупреждение, которое дает вам понять всю серьезность процесса настройки такого типа. **Очень** рекомендуется сделать копии файлов конфигурации. Также неплохо бы иметь под рукой гуру (специалиста по Linux) (просто так, на всякий пожарный).

1.2 Местоположение документа и обратная связь

Этот документ можно найти на sunsite.unc.edu или на tsx-11.mit.edu как часть **Linux Document Project**. Также, его можно найти на различных FTP, имеющих отношение к Linux. Кроме того, это документ может быть включен в дистрибутив Linux как его часть.

Если у вас имеются какие-либо предложения или исправления имеющие отношение к этому документу, то, пожалуйста незамедлительно проконтактируйте со мной по адресу abel@bfr.co.il. За предоставление любой новой и полезной информации относительно поддержки Кириллицы в различных Unixах буду *крайне признателен*. Не забудьте, ведь это поможет другим.

1.3 Благодарность and copyrights

Много людей помогли мне (и не только мне) ценной информацией и предложениями. И даже большое количество людей создало программное обеспечение для public comunity. Мне очень жаль если я забыл кого - то упомянуть.

Итак вот они наши герои: Edward C. Bailey за его неоценимую помощь в создании предметного указателя для этого документа, Bas V. Bakker, Алексей Богданов, Michael Van Canneyt, David Daves, Денис В. Дмитриенко, Vlad Harchev, Дмитрий Малыханов, Сергей О. Наумов, Илья К. Орехов, Winfried Truemper, Сергей Вакуленко, Александр Воробьев, и другие - имя коим "легион хороших людей" из relcom.fido.ru.unix и relcom.fido.ru.linux Usenet групп новостей.

Если у вас имеются вопросы, пожалуйста войдите в контакт с Tim Bynum, координатором Linux HOWTO , по linux-howto@sunsite.unc.edu. Для того чтобы получить номер телефона и дополнительную информацию для контакта вы можете "напустить" на этот адрес finger.

Unix торговая марка X/Open Ltd.; MS-DOS, Windows, Windows 95, and Windows NT торговые марки Microsoft Corp.; X Window System торговая марка X Consortium Inc. Другие торговые марки принадлежат соответствующим владельцам.

Ниже идет оригинальный текст соглашения об авторских правах, которое следует включать во все распространяемые копии этого документа и документов, созданных на его основе.

---

2. Теоретическое обоснование

2.1 Символы и кодировки

Чтобы понимать и печатать символы различных языков, система и программное обеспечение должна быть способна отличить их от других символов. То есть каждый уникальный символ должен иметь уникальное представление внутри операционной

системы, или специфического пакета программ. Такая совокупность всех уникальных символов, которые система способна представить сразу, называется **кодировкой**.

Во время создания большинства операционных системе, никто не позаботился предусмотреть возможность представления информации в программах на других языках, отличных от английского. Поэтому, наиболее популярной кодировкой была (и фактически ей и остается) **ASCII** (Американский Стандартный Код для Информационного Обмена).

Стандарт ASCII (или ASCII с 7ми битами) включает в себя 128 уникальных кодов. Они подразделяются на символы, которые ASCII определяет как, собственно, печатаемые символы, и на так называемые, **символы управления**, которые имели специальные значения в старых протоколах связи. Каждый элемент набора идентифицирован целочисленным символьным кодом (0-127). Подмножество печатаемых символов представляют те, которые находят на клавиатуре пишущей машинки с некоторыми некоторыми добавлениями. Каждый символ занимал 7 младших значимых битов байта, тогда как старший разряд использовался для целей управления (то есть, для управления передачи в старых пакетах связи).

Концепция ASCII с 7ми битами была расширена до ASCII с 8 битами (или **расширенного ASCII**). В этой кодировке, диапазон символов соответствует кодам от 0 до 255. Младшие биты (0-127) - чистый ASCII, в то время как старший разряд добавляет еще 127 символов. Так как эта кодировка обратно совместима с ASCII (символ все еще занимает 8 бит, и коды полностью соответствуют старому ASCII), эта кодировка стал широко использоваться.

Стандарт ASCII с 8 битами не определяет содержание верхней половины таблицы кодировки. Поэтом МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ (ISO) взяла ответственность по определению семейства стандартов, известных как **ISO 8859-X** семейство. Это семейство есть совокупность 8ми битных кодировок, где младшая половина каждой кодировки (символы с кодами 0-127) соответствует ASCII, а старшая половина определяет символы для различных языков. Например, следующие кодовые страницы определены для:

- `8859-0` - Новый европейский стандарт (так называемый **Latin 0**)
- `8859-1` - Европа, Латинская Америка (также известный как **Latin 1**)
- `8859-2` - Восточная Европа
- `8859-5` - Кириллица
- `8859-8` - Идиш

В Latin 1, старшая половина таблицы определяет различные символы, которые - не являются частью Английского алфавита, но присутствует в различных европейских языках (немецкие umlauts, французские диакритические знаки и т.д).

Другая популярная реализация расширенного ASCII - это так называемая **кодовая страница IBM** (названная по имени компьютерной компании, которая создала эту кодировку для своих персональных компьютеров). Эта кодировка в старшей половине содержит псевдо - графические символы.

Программное обеспечение, которое не делает никаких предположений о символах использующих 8-ой бит ASCII данных, называется **чисто 8ми битными**. Некоторые

более старые программы, разработанные для ASCII с 7 битами в памяти, а не для чистых 8ми бит могут работать неправильно с вашими расширенными ASCII данными. Большинство пакетов, однако, способны работать с расширенным ASCII по умолчанию, или для этого требуется простая переконфигурация. Подобные, простейшие программы, требующие какой-либо настройки в этом документе не рассматриваются. Вместо этого я сфокусировал свои усилия на более сложных пакетах.

Для информации о том как создавать *свое* чисто 8ми битное программное обеспечение смотрите раздел Locale зависимое программирование.

Так как в большинстве систем символы описываются 8ю битами, то нет никакого способа расширить ASCII еще больше. Способ создания новых символов в ASCII - это просто создание других расширенных ASCII реализаций. С помощью этого способа кириллица и была введена в ASCII.

Мы уже упомянули стандарт **ISO 8859-5** как тот, в котором определяется кодировки для кириллицы. Но поскольку (как это часто случается со стандартами), он был разработан без принятия во внимание реальных процессов проходящих в СССР (когда это еще было), то единственное, что было действительно достигнуто с введением этот стандарт, так это только увеличение беспорядка с кодировками кириллицы. В то время как, в сообществе Internet очень популярна кодировка **KOI8-R** (см. ниже), **ISO 8859-5** используется большими компаниями, создающими программное обеспечение с возможностью обработки кириллической информации, такое как большие базы данных, решения основанные на базе OpenVMS и т.д.

Другие стандарты для кириллицы включают, так называемую, **Alt** кодировку и кодовую страницу **Microsoft CP1251**. Вышеупомянутый Alt стандарт был разработан нашей "любимой" компанией для MC-ДОС довольно давно. Тогда еще слыхом не слыхивали про сети из IBM PC и поэтому основное усилие состояло в том, чтобы сделать этот стандарт настолько насколько это возможно совместимым с IBM стандартом. Поэтому Alt кодировка - это та самая IBM кодовая страница, где все специфические европейские символы в верхней половине были заменены на кириллицу, оставляя псевдографические символы нетронутыми. Следовательно, это не портило вид программ использующих для работы текстовые окна и также обеспечило символы кириллицы в них. **Alt** стандарт все еще жив и чрезвычайно популярен в среде MC-ДОС.

Microsoft CP1251 кодовая страница - это попытка Microsoft придумать новый стандарт для кодировки кириллицы в Windows. Насколько я знаю, это не совместимо с чем либо еще (и не удивительное, три Ха-Ха)

Ну и наконец: **KOI-8** стандарт. В отличии от **Alt** и **CP1251** он был разработан довольно давно для UNIX машин. Так как UNIX **значит** сеть, то основной идей при создания **KOI-8** стандарта была идея об обеспечении перемещения кириллической информации по сети.

Еще раз вернемся в далекое прошлое. Обычно все работали только со стандартным (7ми битным) ASCII. 8ой бит каждого символа указывал на то, что он или управляющий символ, либо просто мусор. Обычно никто не слал данные чисто 8ми битными (каждый норовил оттяпать 8ой бит у символов). Разработчики **KOI8** применили очень продуманный подход. Они поместили кириллические символы в

верхней части расширенной ASCII таблицы, таким образом, что позиции кириллических символов соответствуют их фонетическим аналогам в английском алфавите в нижней часте таблицы. Это означает, что, если в тексте, написанном в KOI-8, мы убираем восьмой бит каждого символа, *то мы все еще имеем "читабельный" текст, хотя он и написан английскими символами!*

Не удивительно, что **KOI8-R** быстро стал фактически стандартом для кириллицы в Internet. Андрей А. Чернов проделал огромный объем работы, чтобы создать стандарт. Он - автор RFC 1489 (*"Registration of a Cyrillic Character Set"*).

Существуют также и другие стандарты, которые отличны от ASCII и гораздо более хорошо адаптируемы. Наиболее известный из них это **Unicode**. Однако, эти стандарты пока не прижились в Unix вообще и в Linux в частности. Я не описываю их здесь.

---

3. Настройка вашего окружения

Перед тем как мы начнем настраивать различные части системы, нам надо настроить пару простых вещей. Большинство утилит, описываемых ниже, предполагают, что кириллические шрифты доступны и пользователь может вводить кириллические символы. Чтобы это действительно стало правдой, нам следует настроить окружение для того чтобы обеспечить и шрифты, и возможность ввода кириллицы.

Linux поддерживает два интерфейса для отображения информации, две различные моды. Одна из них это текстовая мода, а другая графическая, предоставляемая средствами X Window. Обе эти моды требуют различной настройки, которые описывается ниже.

3.1 Настройка текстовой моды

Вообще, настройка текстовых режимов - самый простой способ показывать и вводить символы кириллицы. Однако, имеется одно значительное осложнение: текстовые шрифты и расположение символов на клавиатуре зависят от реализации драйвера терминала. Следовательно, не имеется никакого общего способа для достижения цели в различных систем.

Ниже, я опишу способ как "справиться" с драйвером Linux консоли. Поэтому если вы имеете другую систему, не ожидайте, что это будет работать. Взамен, проконсультируйтесь с руководством по драйверу терминала и пошлите мне любую информацию, которую вы найдете. В этом случае я смогу включить это в дальнейшие версии этого документа.

Linux консоль

Консольный драйвер Linux - довольно хорошо настраиваемый образец программного обеспечения. Он может менять как шрифты, так и раскладки клавиатуры. Чтобы сделать это, вам нужен пакет kbd. Большинство дистрибутивов Linux устанавливают kbd как обязательную часть системы.

Пакет kbd содержит утилиты управления клавиатурой, кроме этого с ним поставляется широкий выбор шрифтов и раскладок.

Установка кириллицы с помощью `kbd` обычно состоит из:

1. Загрузки соответствующей раскладки клавиатуры, с помощью программы `loadkeys`. Redhat позволяет установить раскладку клавиатуры, которую система загружает по умолчанию во время загрузки. Это конфигурируется с помощью программы `/usr/sbin/kbdconfig`. Или вы просто можете запустить `loadkeys` из вашего `~/.profile` или сделав это руками.
2. Настройки экранного шрифта. Это делается с помощью программы `setfont`. Файлы шрифтов находятся в `/usr/lib/kbd/consolefonts`. **ВНИМАНИЕ:** В старых версиях Linux, запуск программы `setfont` под X Windows мог завесить систему. Сейчас в этом случае печатается сообщение об ошибке.

Если вы являетесь приверженцем программ, выполняемых в текстовой моде, и использующих достоинства PC псевдо - графики (таких как Midnight Commander), вы можите предпочесть использовать шрифты с `Alt` кодировкой и *консольной раскладкой* (console character map). Это означает, что ваша консоль отображает Alt шрифты, но все кириллические символы, соответствующие **KOI-8R** кодировки соответственным образом отображаются в `Alt` и поэтому отображаются правильно. Преимущество этого метода заключается в том, что он позволяет использовать псевдографические символы `Alt` кодировки.

Короче, ниже перечислены команды, которые позволяют достичь этого эффекта.

```
loadkeys /usr/lib/kbd/keytables/ru.map
setfont /usr/lib/kbd/consolefonts/Cyr_a8x16
mapscrn /usr/lib/kbd/consoletrans/koi2alt
echo -ne "\033(K"             # магическая последовательность
```

После выполнения этих команд и загрузки соответствующих файлов, вы можите переключать раскладку клавиатуры для ввода кириллических символов с помощью правого Control.

Магическая последовательность необходима для перекодировки вывода символов на экран если вы используете Alt шрифты. Она работает и вам не следует знать о ней что-то большее. Однако, если вам любопытно, то посмотрите в документацию к пакету kbd.

В заключение, для тех эстетов, кто не желает использовать **Alt** кодировку, я предлагаю другую версию описанной выше загрузочной последовательности, использующей родные **KOI8-R** шрифты.

```
loadkeys /usr/lib/kbd/keytables/ru.map
setfont /usr/lib/kbd/consolefonts/koi8-8x16
```

Однако, не ожидайте красивых рамок в ваших программах, использующих для работы менюшки в текстовом режиме.

Теперь вы, вероятно, хотите проверить это. Сконфигурируйте соответствующим образом bash или tcsh (смотрите ниже этот шаг *необходим*), перегрузите его, затем

нажмите правую клавишу `Control`. Удостоверитесь, что вы можите печатать на русском правильно. Клавиша 'q' должна соответствовать "й", 'w' соответствует "ц", и т.д.

Если у вас возникли непредвиденные проблемы, то лучше всего вернуться к родной (то есть US) раскладке. Для этого сделайте следующие телодвижения:

```
loadkeys /usr/lib/kbd/keytables/defkeymap.map
setfont /usr/lib/kbd/consolefonts/default8x16
```

ВНИМАНИЕ: к сожалению, консольный драйвер не способен сохранить это состояние (по крайней мере без излишних ухищрений), когда передается управление X Windows. Следовательно, после того, как вы вышли из X (или переключаетесь на консоль), то вы должны перезагрузить русский шрифт.

3.2 X Window

Подобно консольному режиму, X Windows также требует некоторой настройки. Настройка включает в себя настройку ввода и установку шрифтов для X Windows. Данные действия обсуждаются ниже.

Шрифты для X Windows.

Прежде всего вы должны достать шрифты, содержащие изображения кириллических символов в соответствующих местах.

В конце 1995 года X Window включила набор **KOI8-R** шрифтов разработанных **KOI8-R** fonts, created by Cronyx. Эти шрифты являются также частью XFree86.

Несмотря на это, некоторые дистрибутивы не включают кириллические шрифты для X Windows в стандартную поставку. Одним из подобных печальных примеров является RedHat (они обещали больше так не делать и исправить это в RedHat 5.2).

Поэтому, вам следует проверить установлены ли эти шрифты в вашей системе. Спросите системного администратора, или, если *вы*- это он и есть, проверьте вашу систему сами, а именно:

1. Выполните '`xlsfonts | grep koi8`'. Если в результате выполнения команды появится список шрифтов, то ваш X сервер уже знает об их существовании.
2. Или, наберите
3. `find / -name crox\*.pcf\*`

   для того чтобы найти местоположение шрифтов кириллицы в системе. Вы должны будете сделать эти шрифты доступными для X сервера, как - я объясню ниже.

Если вы не нашли таких шрифтов в вашей системе, то вы должны установить их сами.

Кроме описанного выше существует еще один набор кириллических шрифтов Cronyx в сети (по адресу ftp.kiae.su), известный, как пакет `xrus` (не путать это с программой

xruskb, ранее известной как xrus. Xrus имеет меньший набор шрифтов чем в коллекции Xfree86 (38 против 68).

Имеется также более старое решение, например пакет vakufonts созданный Сергеем Вакуленко, который стал основой для пакета включенного в дистрибутив X Windows. Очень важно, что имена шрифтов в старой коллекции не совпадают со стандартом полностью. Vakufonts, в общем, неплохой пакет, но иногда могут возникать различные сверхъестественные ошибки. Например, у меня были проблемы с Maple V для Linux, который падал по непонятным причинам с пакетом vakufonts, но прекрасно работал со "стандартными" набором.

Итак, давайте начнем со шрифтов:

1. Вытащите по ftp соответствующий набор шрифтов. Пакет для XFree86 можно найти на любом FTP архиве, где лежит дистрибутив X Windows, например, непосредственно на официальном XFree86 FTP архиве. Пользователи Redhat, у которых отсутствует этот пакет могут заглянуть на ftp.redhat.com, чтобы вытянуть пакет XFree86-cyrillic-fonts. Установить его и сразу перейти к разделу, где объясняется как сделать местоположение шрифтов известным X Windows.
2. Теперь, когда у вас есть шрифты, создайте директорию для них. Поместить новые шрифты в уже существующий каталог шрифтов, мягко говоря, не очень хорошая идея. Поместите их, например, в /usr/lib/X11/fonts/cyrillic для настройки всей системы, или просто создайте каталог у себя только для персонального пользования.
3. Если новые шрифты поставляются в формате BDF (*.bdf файлы), то вы должны скомпилировать их. Для каждого шрифта выполните:
4. `bdftopcf -o <font>.pcf <font>.bdf`

   Если ваш X сервер поддерживает сжатые шрифты, то сожмите их с помощью программы compress (для последних версий XFree86 можно сжать шрифты с помощью программы gzip):

   ```
   compress *.pcf (или gzip *.pcf)
   ```

   Если же вы все-таки хотите поместить новые шрифты в уже существующий каталог шрифтов, то вы должны "срастить" старый и новые файлы, с именем fonts.alias в том случае, естественно, если они оба существуют.

5. В каждом каталоге шрифтов для X должен быть список шрифтов, находящихся в нем. Этот список хранится в файле fonts.dir. Вы не должны создавать этот список вручную. Вместо этого, сделайте:
6. `cd <new font directory>`
7. `mkfontdir .`
8. Теперь Вы должны сделать этот каталог шрифтов известным для X сервера. Здесь у вас есть ряд возможностей:
   - Общесистемная настройка для XFree86. Если вы используете эту версию X Windows, то добавьте новый каталог к списку каталогов в файле

XF86Config. Чтобы найти его расположение, просмотрите что скажет startx при запуске (В Redhat этот файл обычно находится в /etc/X11). Более подробно смотрите man XF86Config (4/5).

- Общесистемная настройка через xinit. Добавьте новый каталог к файлу запуска xinit. Более подробно смотрите xinit(1x).
- Персональная настройка. У вас есть специальный файл для запуска X Windows - ~/.xinitrc (или ~/.Xclients, или ~/.xsession для пользователей RedHat). Добавьте следующие команды в этот файл:

9. xset +fp <новый каталог шрифтов>
10. xset fp rehash

Обратите внимание, на опцию '+fp'- это означает, что новые шрифты будут добавлены в начале списка директорий со шрифтами. То есть, если прикладная программа запрашивает, скажем, шрифт fixed, то будет подставлен fixed font с кириллическими символами , чего мы и добивались. Хотя имеются некоторые проблемы. В дистрибутиве шрифтов кириллицы нет полужирного и курсивного fixed шрифта. Мой любимый шрифт - 6x13, а так, как полужирные и курсивные шрифты этого размера также отсутствуют, то я не могу использовать Emacs/XEmacs в полной мере. Надеюсь, что кто - то в конечном счете создаст эти шрифты, и ситуация изменится.

11. А теперь перезапустите ваш X Windows. Если вы все сделали правильно, то тесты, описанные в начале раздела, будут пройдены успешно. Также, поиграйтесь с xfontsel(1x), чтобы удостовериться что вы способны выбрать шрифты кириллицы.

Чтобы заставить X клиента использовать шрифты Кириллицы, Вы должны установить соответствующие X ресурс. Например, я делаю кириллический шрифт заданным по умолчанию в моем ~/.Xdefaults:

```
*font:          6x13
```

Так как мои шрифты кириллицы первые в списке поиска шрифтов (смотри вывод программы 'xset q'), то они и подставляются если программа требует какой-либо шрифт с именем, совпадающем с каким-либо из шрифтов, лежащих в кириллической директории.

Вот простые примеры. Если Вы хотите научить соответствующий X клиент пользоваться кириллическим шрифтом , то вы должны узнать тип ресурса , который вам надо изменить (можно это сделать используя editres(1x)) и определить его или в базе данных ресурсов, или в командной строке. Например можно сделать так:

```
$ xterm -font '-cronyx-*-bold-*-*-*-19-*-*-*-*-*-*-*'
```

...запусти xterm с несколько уродливым шрифтом;

```
$ xfontsel -xrm '*quitButton.font: -*-times-*-*-*-*-13-*-*-*-*-*-koi8-*'
```

...установит Cyrillic Times шрифт для кнопки `Quit` программы `xfontsel`.

Использование шрифтов TrueType

У технологии TrueType, взятой на вооружение операционными системами серии Windows (он же Mastdie :)), нет многих недостатков, которые присутствуют при работе со шрифтами стандартным образом у X. Кроме того существует множество кириллических TrueType шрифтов.

К счастью, эти шрифты можно использовать и в X Windows. Существует TrueType сервер шрифтов XFSFT для X (говорят, что в скором времени поддержка TrueType будет включена в дистрибутив XFree86).

Установочная процедура довольно проста - просто, делайте то, что сказано в документации. Следует отметить одну вещь, имеющую отношение к кириллической настройке, это то, что вам следует использовать опцию перекодировки:

```
xfstt ... --encoding koi8-r,windows-1251,iso8859-1
```

После этого, новые шрифты будут доступны для X Windows - этот факт вы можете проверить, запустив `xfontsel` и поиграв с *ttf* семейством шрифтов.

Ввод с клавиатуры

В последних выпусках X (X11R61 и выше) имеются два "стандартных" способа ввода с клавиатуры: родной способ, работающий через утилиту `xmodmap`, и новый, называемый **Xkb** (X KeyBoard). Первое, что вы должны сделать, это **отключить Xkb способ!**. Не давайте себя загипнотизировать его способностью устанавливать "russian keyboard". Этот способ, вроде бы, использует описание символов кириллицы определенных в `keysymdef.h`. Этот файл определяет символы для многих языков. Единственная проблема состоит в том, что данное определения не позволяют, работать с расширенной ASCII кодовой страницей - очень много программ могут работать только с этой раскладкой! Я почти не знаю программ, которые разбираются с описанием символов в `keysymdef.h` отличным от стандартного 8ми битного ASCII. Однако, наша главная цель состоит в том, чтобы заставить работать поддержку KOI8-R. (Примечание переводчика: здесь автор немного погорячился - Xkb вполне рабочий способ, правда способность работать с KOI8-R он обнаружил только в последних версия XFree86. Пользователям Redhat следует взять версию XFree86-3.3.3-1.)

Чтобы отключить поддержку `Xkb`, в файле `XF86Config` изучите раздел `Keyboard` и закоментируйте, все строки, начинающиеся с **Xkb** (регистр не имеет значения). Взамен, добавьте следующую строку:

```
XkbDisable
```

Программа xmodmap позволяет настроить коды соответствующие различными символам и их комбинациям. Эта настройка основана на файле, содержащем таблицу перекодировки.

В предыдущих версиях этого документа я очень подробно описывал настройку кириллицы основанную на xmodmap. Это оказалось практически бесполезным. Общеизвестно, что способ перекодировка ввода, основанный на xmodmap, является, не-портабельным, плохо настраиваемым, и не окончательным. Ваши настройки могут работать с одной версией XFree и сбоить при работе с другой. Более того, иногда результат работы одной и той же настройки сильно разнится для различных серверов из одного дистрибутива.

Я настоятельно советую вам не играть с xmodmap, по крайней мере для начала. Вы не получите ничего кроме головной боли и разочарования. Вместо этого, я рекомендую вам установить пакет xruskb, который позволяет вам сконфигурировать большинство входных параметров перекодировки без необходимости сталкиваться с xmodmap. Опять же пользователи Redhat Linux могут установить RPM пакет этого програмного продукта.

3.3 Первые шаги - кириллизация shell

bash

Для того, чтобы заставить bash понять 8ми битные символы, должны быть установлены три переменные. Лучше всего это сделать из файл ~/.inputrc. Должны быть сделаны следующие установки:

```
set meta-flag on
set convert-meta off
set output-meta on
```

csh/tcsh

Поместите в .cshrc следующее:

```
setenv LC_CTYPE iso_8859_5
stty pass8
```

Если у вас нет POSIX совместимого stty (только не в Linux), замените последнюю строчку следующей:

```
stty -istrip cs8
```

ksh

Что касается public domain реализации ksh - pdksh версии 5.1.3 и выше, то вы можете разрешить 8ми разрядный ввод только для vi в режиме ввода. Для этого используйте:

```
set -o vi
```

less

Установите переменную окружения `LESSCHARSET`:

```
export LESSCHARSET=koi8-r
```

Старые версии `less` не поддерживали символы KOI8-R, но установка следующей системной переменной позволяла это обойти:

```
export LESSCHARSET=latin1
```

mc (The Midnight Commander)

Чтобы разобрать текст кириллицы , выберите опцию `full 8 bits` в `Options/Display` меню.

Если у вас проблемы в виде уродливых оконных рамок, то проконсультируйтесь в разделе Linux консоль.

off-topic: если вы захотите чтобы `mc` в окне `Xterm` был в цвете: установите переменную `COLORTERM`:

```
COLORTERM= ; export COLORTERM
```

rlogin

Удостоверитесь, что shell на месте адресата правильно установлена. Если ваш `rlogin` не работает как надо по умолчанию, то используйте '`rlogin -8`'.

zsh

Сделайте то же самое, что делается для `csh` (csh)*. Файл инициализации в этом случае - `.zshrc` или `/etc/zshrc`.

---

* csh/tcsh

Поместите в `.cshrc` следующее:

```
setenv LC_CTYPE iso_8859_5
stty pass8
```

Если у вас нет POSIX совместимого stty (только не в Linux), замените последнюю строчку следующей:

```
stty -istrip cs8
```

---

4. Редактирование текста

В этом разделе я опишу настройку различных текстовых редакторов для работы с кириллическими текстами. Этот раздел на охватывает **текстовые процессоры**, настройка которых описывается позже (смотри раздел Текстовые процессоры)

4.1 Emacs и XEmacs

Есть две версии редактора Emacs - `GNU Emacs` и `XEmacs`. Они обеспечивают более или менее сходный набор функций и возможностей. Реализация некоторых деталей расходится довольно сильно. Настройка кириллицы требует некого низко-уровневого (в духе Emacs Лиспа) "хака" и немного отличается для этих двух реализаций.

ВНИМАНИЕ:Помимо настройки, описанной здесь, существует альтернативный путь обучения обоих версий emacs кириллице. Для этого используется `MULE` (MULtilanguage Emacs support). Этот путь несколько сложен и (на сколько я знаю) редко используется, поэтому я его здесь не описываю.

Минимальная поддержка кириллицы в `GNU emacs` (вам не нужно этого делать при настройке `XEmacs`) обеспечивается при выполнении следующих вызовов, добавленных в `.emacs` (при условии, что поддержка символов кириллицы установлена для консоли или для X Windows соответственно):

```
(standard-display-european t)

(let ((m (current-input-mode)))
  (set-input-mode (car m) (nth 1 m) 1))
```

Это позволит вам видеть и вводить символы, находящиеся в верхней части ASCII таблицы.

Однако, этого не достаточно. Emacs обрабатывает кириллические символы как специальные, и как следствие не распознает границы русских слов и не делает различия между строчными и заглавными буквами. Чтобы обойти это, вам следует несколько модифицировать таблицы синтаксиса и регистра emacs:

```
(require 'case-table)

(let* ((ruc
"\341\342\367\347\344\345\263\366\372\351\352\353\354\355\356\357\360\362\3
63\364\365\346\350\343\376\373\375\370\371\377\374\340\361")
       (rlc
"\301\302\327\307\304\305\243\326\332\311\312\313\314\315\316\317\320\322\3
23\324\325\306\310\303\336\333\335\330\331\337\334\300\321")
       (i 0)
       (len (length ruc)))
  (while (< i len)
    (modify-syntax-entry (elt ruc i) "w  ")
    (modify-syntax-entry (elt rlc i) "w  ")
    (set-case-syntax-pair (elt ruc i) (elt rlc i) (standard-case-table))
    (setq i (+ i 1))))
```

Для этого я создал файл `rusup.el`, который содержит эти команды, также как и пару других удобные функции. Вы должны вызвать его в вашем `~/.emacs`.

Ну и в заключении: пакет russian.el, созданный Валерием Алексеевым (`valery@math.uga.edu`), позволяющий пользователю переключаться между кириллическим и стандартным модами ввода и преобразовывать текстовый буфер из одной русской кодировки в другую (это очень полезно, для чтения текстов импортированных из MS-DOS или Windows).

4.2 Работа с vi

Редактор `vi` (по крайней мере его клон `vim`, присутствующий в большинстве дистрибутивов Linux) знает о существовании 8ми битных символов. Это дает вам возможность вводить кириллические буквы. Редактор правильно распознает границы слов. Я ничего не знаю по поводу правил преобразования из строчных в заглавные и обратно, так как я не часто работаю в `vi`. *Если вы знаете что-либо об этом, то пожалуйста сообщите мне*

4.3 Редактирование текста в joe

Для того чтобы распознавать 8ми битные символы `joe` требует специальную опцию `-asis`. Вы можете указать ее в командной строке или вставить в файл `~/.joerc` для личного пользования или в `/usr/lib/joerc` для настройки всей системы.

Если ваша программа не воспринимает `-asis`, то вам следует обновить ее версию.

Однако, `joe` не распознает границы русских слов. Я предполагаю, что тоже самое происходит и с перекодировкой из верхнего регистра в нижний и обратно.

4.4 Проверка правописания на русском языке

Для проверки правописания я использую программу `GNU ispell`. Она имеет очень гибкие настройки и возможности для расширения. Ее можно использовать при проверки правописания текстов написанных на языках отличных от английского путем добавления новых **словарей**.

Константин Книжник создал *очень* хороший русский словарь для `ispell`. Вы можете найти его на Домашней страничке К.Книжника. В поставку словаря включен полезный скрипт, обеспечивающий инкрементальный режим проверки правописания слов для `emacs`;

В идеальном случае ваш `ispell` установлен правильно, и вам надо только создать словарь, используя команды, обеспечиваемые файлом `Makefile` из дистрибутива. Однако с довольно большой вероятностью у вас возникнут проблемы с `ispell`, который откажется понимать 8ми битные данные. Это может произойти по той причине, что в большинстве дистрибутивов Linux `ispell` скомпилирован без поддержки 8ми битных символов. В этом случае вам не удастся избежать перекомпиляции пакета `ispell`.

К счастью пользователей RedHat я скомпилировал пакет `ispell` вместе с русским и немецким словарями. Как обычно, вы можете утянуть его с RedHat FTP site.

Если вы все сделали правильно, то можете инициировать проверку правописания для русских текстов, путем вызова программы `ispell` с опцией `'-d russian'`

Теперь, если вы используете `Emacs`, то вероятно вы не против добавить пункт в меню для проверки с русским словарем. Я послал соответствующие изменения к человеку, поддерживающему `ispell.el`, и он согласился включить его в файл при следующем

официальном релизе. Кроме того, вы можете сделать это же путем добавления следующего кода в ваш `~/.emacs` (или в `/usr/share/emacs/site-lisp/site-start.el` для настройке всей системы)

```
(setq ispell-dictionary-alist
  (append ispell-dictionary-alist
          '(("russian"

"[\341\342\367\347\344\345\263\366\372\351\352\353\354\355\356\357\360\362\
363\364\365\346\350\343\376\373\375\370\371\377\374\340\361\301\302\327\307
\304\305\243\326\332\311\312\313\314\315\316\317\320\322\323\324\325\306\31
0\303\336\333\335\330\331\337\334\300\321]"

"[^\341\342\367\347\344\345\263\366\372\351\352\353\354\355\356\357\360\362
\363\364\365\346\350\343\376\373\375\370\371\377\374\340\361\301\302\327\30
7\304\305\243\326\332\311\312\313\314\315\316\317\320\322\323\324\325\306\3
10\303\336\333\335\330\331\337\334\300\321]"
             "[']" t ("-C" "-d" "russian") "~latin1"))))

(define-key-after ispell-menu-map [ispell-select-russian]
  '("Select Russian (KOI-8)" . (lambda ()
                                 (interactive)
                                 (ispell-change-dictionary "russian")))
  'british)
```

К сожалению, это не работает в `XEmacs`. Я попытаюсь разрешить эту проблему позже.

---

5. Использования кириллицы в программах электронной почты и чтения новостей

Настройка программ электронной почты и чтения новостей для распознавания кириллицы не очень сложна, хотя вам следует знать основные принципы работы электронной почты и новостей.

Обычно программы Internet электронной почты состоят из двух частей: **MUA** (Mail User Agent - пользовательская программа электронной почты) и **MTA** (Mail Transfer Agent - программа рассылки электронной почты). MUA - это программа, которую вы используете для чтения, составления и посылки электронных сообщений. Однако, пользовательская программа электронной почты сама письма не посылает, вместо этого она вызывает программу рассылки электронной почты, которая отвечает за посылку сообщения, в соответствующем направлении, используя соответствующий протокол. Пример пользовательской программы электронной почты - `Pine`, программу рассылки электронной почты - `qmail`.

До недавнего времени и MTA, и MUA по умолчанию не были чисто 8ми битными программами. Поэтому, когда вы посылаете сообщение, скажем из Америки в Россию, то вы никогда не будете уверены в том, что какая - нибудь промежуточная программа рассылки электронной почты не "откусит" в вашем сообщении 8ой бит у каждой буквы в вшем сообщении. Поэтому был разработан ряд протоколов, которые позволь закодировать любой тип данных с использованием только печатаемых символов из 7ми битного ASCII. Эта группа протоколов зовется **MIME** (MultimedIa Mail Encoding - мультимедийная кодировка электронных сообщений)

Так как MIME обычно сконфигурирован по умолчанию довольно правильно, то мы не будем обсуждать его здесь. Мы поговорим MIME, когда будет обсуждаться совместимость между русскими кодировками (См. раздел mime).

Мы начнем с настройки пользовательской программы электронной почты, потому что с этой программой вы работаете непосредственно. Затем мы обсудим простейшие принципы конфигурации MTA для поддержки кириллицы.

5.1 Настройка пользовательской программы электронной почты (MUA)

Пользовательской программа электронной почты на основе Emacs

Если вы уже настроили emacs как таковой, то вам не нужно специально настраивать программу электронной почты, которая обеспечивается средствами Emacs. (Для настройки Emacs смотрите раздел Emacs и XEmacs).

elm

Добавьте следующую запись в ваш `~/.elm/elmrc`:

```
CHARSET=koi8-r
```

pine

Добавьте следующую запись в `~/.pinerc` для персональной настройки или в `/usr/lib/pine.conf` для настройки всей системы.

```
character-set=koi8-r
```

Вы можете также изменить настройку вашего `Pine` для того, чтобы предотвратить посылку письма в `quoted-printable` кодировке.

```
enable-8bit-nntp-posting
enable-8bit-esmtp-negotiation
```

Кроме того, удостоверьтесь, что вы имеете свежую версию `pine`. До недавнего времени эта программа имела различные проблемы с русским кодировками.

5.2 Настройка вашей программы рассылки электронной почты (MTA)

В "природе" существует несколько программ рассылки электронной почты (MTA) доступных для вас. Это `sendmail`, `qmail`, `smail`, `exim`, и так далее.

sendmail

Не так давно программа `sendmail` была гораздо более популярной чем другие программы рассылки, потому что имела долгую историю и, соответственно, имела

широкое распространение. Что касается меня лично, то я ненавижу эту программу - это прекрасный пример абсолютно бездумного подхода к созданию программного обеспечения и даже "улучшения", которые делаются время от времени, показывают что данный подход не умирает. Любой системный администратор вздрагивает, когда слышит зловещее слово "`sendmail.cf`" (Примечание переводчика - данное мнение было выражений личных привязанностей автора, существуют и другие мнения на этот счет. На данный момент все остальные программы рассылки, как правило, либо гораздо хуже, либо не отвечают запрашиваемым требованиям).

В данный момент `sendmail` больше не откусывает 8мой бит. Однако он может *закодировать* 8ми битные данные с помощью специального **base64** формата кодировки. Хотя большинство программ чтения электронных сообщений распознают и декодируют подобные сообщения обратно в 8ми битный текст, вы вероятно хотите посылать просто текст и быть уверенным, что все работает нормально.

Начиная с версии 8, `sendmail` обрабатывает 8ми битные данные по умолчанию правильно. Если этого не происходит- проверьте опцию `EightBitMode` и опцию `7` в разделе mailers в вашем файле `/etc/sendmail.cf`. Подробнее смотрите руководство *"Sendmail. Operation and Installation Guide"* .

Другие программы рассылки электронной почты

Я немного знаю о других программах рассылки почты. Если вы что-то знаете, что может быть важно для настройки кириллицы, то пожалуйста сообщите мне.

---

6. Путешествие по русскому WWW

В отличии от программ электронной почты и чтения новостей нет никаких стандартов для русской кодировки на WWW. Основная причина заключается в том, что всеми "любимая" компания Microsoft предоставляет программы создания Web сайтов, которые знают о существовании только одной русской кодировки **cp1251**, полностью игнорируя существование других стандартов.

Настройки, описанные здесь, очень примитивны. Они позволят вам видеть страницы в **KOI8-R** кодировке. Если ситуация как-то изменится, то я добавлю еще информации.

6.1 lynx

Начиная с версии 2.6, вы можете выбирать соответствующее значение для дисплея - `display Character set`. (Прим. переводчика: lynx версии 2.8 позволяет смотреть страницы в любой кодировке, а не только в **KOI8-R**)

6.2 Netscape Navigator

Убедитесь, что вы используете `Netscape` версии 4.06 и выше. Начиная с этой версии `Netscape` поддерживает кириллицу гораздо лучше.

Основные настройки

Наконец то, свершилось, теперь Netscape поддерживает русские кодировки правильным образом. Вам надо только правильно настроить шрифты KOI8-R. После этого, если например требуется показать документ, который использует CP-1251, Netscape автоматически перекодирует весь документ в KOI8-R и правильно отобразит его, даже если у него нет доступа к CP-1251 шрифтам.

Для того чтобы правильно настроить KOI8-R в вашем Netscape, сделаете следующее:

- В меню `Options/General Preferences/Fonts` выберите `Cyrillic (KOI-8)` кодировку.
- Выберете подходящие шрифты для этой кодировки - например выберете `Times(Cronyx)` - как пропорциональный шрифт и `Courier(Cronyx)` - как fixed.
- Сохраните настройки.

ВНИМАНИЕ: Все больше и больше появляется WWW страниц, оформление которых сильно зависит от определенных шрифтов. Это в основном касается страниц, созданных под и для MS Windows. Я настоятельно рекомендую установить *сервер TrueType шрифтов*. С помощью него некоторые странички станут выглядеть гораздо лучше. Для подробностей смотрите раздел Использование шрифтов TrueType

Некоторые WWW странички в интернете правильно отвечают на запрос по поводу используемой при их создании кодировки. Другие требуют, чтобы вы выбрали кодировку сами. Для того чтобы это сделать выберете правильную опцию кодировки в меню `Options/Document Encoding`.

Netscape версии 4.08 правильно отображает элементы форм, использующих правильную кодировку (по крайней мере для большинства страниц, которые я видел). Более старые версии делают это неправильно. Однако, если вы по каким-то причинам вам надо использовать более старую версию, или правильное отображение форм по какой-либо причине не работает, то попробуйте сделать следующее:

1. Скопируйте базу данных установок Netscape (обычно `Netscape.ad`) в `~/Netscape`
2. В файле, установите следующую опцию:
3. `*documentFonts.charset*iso8859-1:  koi8-r`

Это вынудит все фреймы и элементы ввода использовать шрифты с кодировкой **koi8-r** вместо заданных по умолчанию, а следовательно вы должны удостовериться в том, что вы уже установили такие шрифты (см. раздел xfonts).

Плохая новость, об использовании этого трюка заключаются в том, что если вы загружаете документ, который должен отображаться с помощью шрифтов `iso-8859-1`, то вместо этого он отобразится с помощью `koi8` шрифта. Иногда такие документы выглядят хуже.

Если вам нужно чего-то еще: Андрей А. Чернов - это человек который знает больше чем другие о KOI-8 в общем и в netscape в частности. Посетите его превосходную KOI-8 page страницу, и скачайте заплату для файла ресурсов Netscape, который заставляет Netscape говорить по Русски, так хорошо как это только возможно.

7. "Русскоязычные" текстовые процессоры

7.1 Поддержка кириллицы в TeX

В этом разделе я опишу несколько способов набора русских текстов в TeX и LaTeX. Есть несколько путей для достижения этой цели, которые отличаются в сложности установки и удобстве использования. Например, одна из возможностей это начать работу без всякой предварительной настройки, используя *Washington AMSTeX Cyrillic fonts*. С другой стороны, вы можете установить пакет LaTeX, который легко настраивается на пользование кириллицей. У меня был опыт работы с двумя подобными пакетами. Один из них - пакет `cmcyralt` создан Вадимом В. Житником (`vvzhy@phy.ncu.edu.tw`) и Александром Харином (`harin@lourie.und.ac.za`), а другой - пакет `LH` написан группой **CyrTUG**. Пакет `LH` включает в себя стили и расстановку дефисов для LaTeX2e, созданную Сергеем О. Наумовым (`serge@astro.unc.edu`). Я опишу оба.

Обратите внимание, что доступны две версии LaTeX, одна из них - 2.09 - старая версия , в то время как 2e - новая (выпуск pre-3.0). Если Вы используете LaTeX 2.09, то как можно быстрее переходите на 2e. Последний сохраняет совместимость со старой версией, но имеет намного больше возможностей. Кроме того, версия 3 будет скоро выпущена. Я опишу установку LaTeX 2e.

Да, кстати, оба этих пакета требуют для редактирования русских текстов установку **Alt** кодировки, а не **KOI8-R**! Это вызвано историческими причинами, создатели этих пакетов, использовали, их для работы с `EmTeX`- MS-DOS версии TeX (они еще знали о Linux :-). Переход к **KOI8-R** требует некоторых усилий и ожидается, что будет скоро сделан. А пока, используйте какие-нибудь утилиту, для перекодировки русского текста из **KOI8-R** в **Alt**. См. раздел Символьная перекодировка.

Работа с Washington Cyrillic

Этот пакет был создан для Американского Математического Общества, чтобы дать возможность создавать документы со ссылками на первоисточники на русском. Следовательно, авторы не очень "напрягались" при создании этого пакета и шрифты в результате этого выглядят довольно неуклюже. Обычно этот пакет упоминается как "`по настоящему плохой пакет кириллицы для TeX`".

Однако, мы обсудим его, так как он очень прост в использовании и не требует установки - этот набор содержится в большинстве дистрибутивов TeX.

Конечно, у вас не будете такой роскоши как автоматическая расстановка дефисов, но все равно ...

1. Снабдите ваш документ следующими директивами:

```
\input cyracc.def
\font\tencyr=wncyr10
\def\cyr{\tencyr\cyracc}
```

2. Теперь, чтобы напечатать символы кириллицы, вставьте

```
\cyr
```

для печати используйте соответствующий латинский символ или команду TeX. То есть, строчные буквы русского алфавита соответствуют следующим сочетаниям:

```
a b v g d e \"e zh z i {\u i} k l m n o p r s t u f kh c ch sh shch
{\cprime} y {\cdprime} \`e yu ya
```

Чрезвычайно сложно преобразовывать ваши русские тексты в такую кодировку, но вы можете автоматизировать этот процесс. Программа translit (раздел Символьная перекодировка) поддерживает опцию вывода TeX.

Пакет KOI-8 для teTeX

Также есть довольно новый пакет teTeX-rus. Он поддерживает набор символов KOI-8 и отвечает всем основным требованиям TeX и LaTeX. Я лично его не пробовал, но я слышал что его успешно используют.

ВНИМАНИЕ: Этот пакет требует, чтобы вы переконфигурировали и пересобрать некоторые части вашего пакета `teTeX` (например скомпилированные для дальнейшего использования LaTeX макрокоманды). *Если вы не знаете точно, что делаете, то вы не должны пробовать это без некой разумной доли осторожности. Вероятно лучше позаимствовать правильно пересобранные части у кого - либо из сети.*

Использование пакета для LaTeX cmcyralt

Пакет cmcyralt может быть найден на любом архиве CTAN (Comprehansive TeX Archive Network)- например на `ftp.dante.de`. Вы должны получить две его части: набор шрифтов из `fonts/cmcyralt` и набор стилей с правилами расстановки дефисов, которые находятся в директории `macros/latex/contrib/others/cmcyralt`.

ВНИМАНИЕ: Удостоверитесь, что у вас установлен пакет `Sauter`, так как `cmcyralt` требует наличия некоторых шрифтов из этого пакета. Этот пакет вы также можете достать из любого архива CTAN.

Теперь Вы должны сделать следующее:

1. Поместите новые шрифты в каталоги шрифтов TeX. На моей системе (Slackware 2.2) я создал каталог `cmcyralt` в `/usr/lib/texmf/fonts/cm/`. Создайте подкаталоги `src`, `tfm`, и `vf` в нем. Поместите там `.mf`, `.tfm`, и `vf` файлы соответственно.
2. Поместите файлы драйвера шрифтов (`*.fd`) из набора стилей в соответствующее место (в моем случае это было в `/usr/lib/texmf/tex/latex/fd`).
3. Поместите файлы стиля (`*.sty`) в соответствующий каталог стилей LaTeX (в моем случае это было в `/usr/lib/texmf/tex/latex/sty`)).

Теперь настройка расстановок переносов. Она потребует, пересобрать основные файлы LaTeX.

1. Файл `hyphen.cfg` содержит директивы для и английской и русской расстановки переносов. Извлеките директивы для русских переносов, и поместите их в файл конфигурации расстановки переносов LaTeX - `lthyphen.ltx`. В моем случае, этот файл был в директории `/usr/lib/texmf/tex/latex/latex-base`.
2. Поместите `rhyphen.tex` в тот же самый каталог. Это необходимо для создания основного файла. Позже, вы можете удалить его.
3. Выполните команду "make" в этом каталоге. Не забудьте сделать линк от `Makefile` к `Makefile.unx`. Во время компиляции следите за выводом. Должно быть сообщение:
4. `Loading hyphenation patterns for Russian.`

   Если все прошло O'K, то вы получите в этом каталоге новый `latex.fmt`. Поместите его туда, где был предыдущий (вероятно в `/usr/lib/texmf/ini/`). **Не забудьте сохранить предыдущую версию файла (так, на всякий пожарный)!**.

Это все. Установка закончена. Пробуйте пройтись по примерам, найденным в архиве стилей. Если вы в состоянии создавать PostScript файлы без каких-либо проблем, то все - OK. Теперь, чтобы использовать кириллицу в LaTeX, снабдите ваш документ следующей директивой:

```
\usepackage{cmcyralt}
```

Для подробностей, смотрите `README` файл в архиве стилей `cmcyralt`.

Обратите внимание: если у вас есть проблемы с примерами, и если вы все сделали правильно, то вероятно ваш пакет TeX не был правильно установлен. Например, во время моей первой попытки, каждая попытка создать `.pk` файлы для русских шрифтов терпела неудачу (стадия `MakeTeXPK`). Пристрастное расследование выявило некий неявный конфликт между *localfont* и *ljfour* `METAFONT` настройками. Прежде это работало, но терпело крах после установки cmcyralt. Войдите в контакт вашим местным TeX гуру. TeX очень (иногда слишком) сложен, чтобы переконфигурировать его без отсутствия навыков.

Использование пакета CyrTUG

Вы можете найти пакет CyrTUG в архиве SunSite. Возьмите файлы `CyrTUGfonts.tar.gz`, `CyrTUGmacro.tar.gz`, и `hyphen.tar.Z`.

Процесс установки не слишком отличается от предыдущего.

7.2 StarOffice

Юрий Коваленко (http://www.inp.nsk.su/~kovalenko) собрал всю информацию по русификации StarOffice. Она находится по адресу ftp://sky.inp.nsk.su/archives_src/linux/StarOffice/russification.txt. У меня не было возможности попробовать это и я ничего не могу сказать о точности этой информации.

Другой источник информации по этому вопросу предоставлен Евгением Демидовым (mailto:jack@gpi.ru) и расположен по адресу ftp://ftp.kapella.gpi.ru/pub/cyrillic/psfonts/README.

Ну и наконец, Star Division Corp. планирует организовать поддержку кириллицы в грядущей версии StarOffice 5.0

---

8. Вывод на печать and PostScript

Напечатать что-либо - это всегда проблема. Имеется набор различных принтеров от различных производителей с различными особенностями. Даже для вывода на печати обычного ASCII текста нет никаких общих решений (это применимо не только к UNIX, но также и к другим операционным системам.

Принтеры имеют различные управляющие языки, и очень часто они имеют сильно различные подходы к поддержке иностранных языков. Хорошая новость - это то, что сейчас в качестве управляющего языка, как стандарт de facto для описания работы печати используется язык **PostScript**, разработанный Adobe Corporation. Много принтеров имеют встроенный **PostScript интерпретатор**, то есть вам надо просто послать на принтер Postscript данные. Для тех у кого этого нет, существует **программные PostScript интерпретаторы**, который берет данные Postscript преобразовывает в специфический для данного принтера управляющий код. Один из них мы сейчас обсудим (вероятно самый лучший из лучших). Это GNU GhostScript (gs для краткости).

Другая проблема - это широкий спектр требований предъявляемый к печати. Например, иногда вы хотите просто напечатать часть вашей C программы, содержащей в качестве комментария текст на русском, так что вы не нуждаетесь в "навороченной" процедуре печати- вам нужен простой ASCII вывод с одним шрифтом. Совсем другой дело, когда вы создаете открытку для вашей подруги. В этом случае вы, вероятно, будете нуждаться в печати документа с различными шрифтами и т.д. И это уже определенно требует больших усилий по установке поддержки кириллицы.

Чтобы выполнить вышеупомянутую задачу по выводу C программы, вы должны заставить ваш принтер понять только *один* шрифт кириллицы и (возможно) устанавливать некоторую программу- фильтр, чтобы выводить данные в соответствующем формате. Чтобы совладать со второй задачей, вы должны обучить ваш принтер различным шрифтам и иметь специальное программное обеспечение.

Бывают задачи, требующие для выполнения нечто среднее, тогда вам нужна программы, которая знает, как организовать и шрифты, и соответствующий вывод в принтер, так что вы можете, скажем, получить на выходе качественно отпечатанный текст, без сложных систем подготовки текстов.

8.1 Преобразование текста в PostScript

Иногда у вас есть простой ASCII KOI-8 текст, и вы хотите его только напечатать. Один из самых простых способов это сделать это воспользоваться услугами программ преобразующих текст в PostScript.

Есть ряд программ, делающих такое преобразование. Я лично предпочитаю a2ps. Первоначально разработанная как простой text-to-PostScript преобразователь, эта программа сейчас стала "матерым", легко настраиваемым и с большим количеством опций программным продуктом. Она позволяет управлять форматами, размещениями страниц, выделением и т.д. Другая утилита, делающая примерно то же самое (теперь доступная как часть проекта **GNU**) - enscript.

A2ps конвертер

Преобразователь текста в PostScript был и остается одним из наиболее универсальных средств печати. Автор, как оказалось, очень открыт для предложений, и как следствие, `a2ps` версия 4.9.8 поддерживает кириллицу прямо в программе. Все, в чем вы теперь нуждаетесь - это PostScript принтер.

Команда, которую я использую для этого:

```
a2ps -X koi8r --print-anyway  <файл>
```

GNU enscript

Программа GNU `enscript` также, как и a2ps, была разработана для преобразования текста в PostScript, и она также поддерживает не ascii кодировки. Программа не имеет русских PostScript шрифтов в своем составе, но их очень просто доустановить. Как это сделать описано ниже (спасибо Michael Van Canneyt):

1. Установите последний `enscript`. Теперь, самая последняя версия это 1.5. вы можете найти ее на GNU FTP архиве, или взять пакет RPM с Redhat.
2. Если вы - счастливый пользователь RedHat Linux, загрузите и установите шрифт Cyrillic Textbook.
3. Если вы не используете RPMки, вытяните файл `textbook.tar.gz` из архива на sunsite.unc.edu (сдесь лежит программное обеспечение для русификации). Разархивируйте этот файл в каталог, где размещены шрифты для `enscript` (обычно `/usr/share/enscript`). Теперь перейдите туда, и выполните следующую команду:
4. `mkafmmap *.afm`
5. Установка завершена. Попробуйте напечатать текст в KOI8-R следующей командой:
6. `enscript --font=Textbook8 --encoding=koi8 some.file`

Если вам требуется действительно быстрый и простой способ, и качества вывода для вас не критично, и все что вам нужно - это только русский текст на бумаге, попробуйте пакет rtxt2ps. Это очень простой без украшательств конвертер текста в PostScript. Качество вывода не очень хорошее (или, честно говоря *плохое*) но это - работает.

8.2 Преобразование текста в TeX

Если все, в чем вы нуждаетесь- это печатать ASCII текста без дополнительной обработки, то вы можете использовать некоторые программки, которые могут преобразовать ваш текст кириллицы в готовый TeX файл. Одна из самых лучших программ для таких целей - это `translit` (Смотрите раздел Символьная перекодировка). В этом случае вы даже не должны беспокоиться относительно установки шрифтов кириллицы для TeX, так как `translit` использует пакет кириллицы `Washington Cyrillic`, который включен в большинство дистрибутивов TeX (или - я не прав?).

8.3 Кириллица в PostScript

Эксперты говорят, что PostScript это просто. Я не могу судить - у меня было слишком много вещей, которые надо изучить, чтобы выкроить время для изучения PostScript. Но я все равно попробую использовать мои небольшие знания об этом вопросе. **Я буду очень благодарен за любую информацию об этом вопросе от вас друзья, которые знают больше меня** (вас приблизительно 99 % от Земной популяции).

Чтобы печатать русский текст с использованием PostScript, вы должны удостовериться относительно следующих вещей:

- шрифт кириллицы *загружен* или включен в документ.
- текст кириллицы включен в документ.
- текст кириллицы использует соответствующие символьные коды, которые соответствуют требованиям шрифта.
- чтобы печатать текст кириллицы *выбран* соответствующий шрифт.

Не имеется никакого достаточно общего решения, чтобы рекомендоваться его как окончательное. Я попробую осветить различные способы для решения различных проблем, связанных с этим вопросом.

Один способ это побороть проблемы установки кириллицы вообще, состоит в том, чтобы использовать Ghostscript. `Ghostscript` (или просто `gs` от newspeak) абсолютно free (ну, не совсем) интерпретатор PostScript. У него есть много преимуществ; среди них:

- Способность работать на многих платформах (различные Unix, Окошки и т.д)
- Поддержка для огромного количества не-PostScript принтеров
- Высокая степень настраиваемости

В нашем специфическом случае является важным то, - что однажды установив и настроив `Ghostscript`, мы можем все печатать через него, таким образом нам не надо дополнительно настраивать другие PostScript устройства (например **HP LaserJet IV**).

Добавление шрифтов кириллицы к Ghostscript

Это важно, так как вы, вероятно, не захотите взваливать ответственность за включение шрифтов кириллицы в PostScript на другие программы. Взамен, вы добавляете их только к gs и заставляете программы выводить русский текст совместимый с этими шрифтами.

Чтобы добавлять новый шрифт (в формате `pfa` или `pfb`) в `gs`, вы должны:

1. Поместить этот шрифт в каталог шрифтов tt/gs/ (то есть. в `/usr/lib/ghostscript/fonts`).
2. Добавить соответствующие имена и aliases для шрифта в файле `Fontmap` в каталоге `gs`.

Недавно появился приличный набор шрифтов кириллицы для `GhostScript`. Его можно найти на ftp.kapella.gpi.ru. Этот набор даже имеет необходимую часть для добавления к файлу `Fontmap`. Вы должны утянуть содержание каталога `/pub/cyrillic/psfonts`. В файле README описываются все необходимые подробности.

8.4 Использование старого матричного принтера для печати кириллического текста

Если у вас есть старый, добрый матричный принтер, и вы нуждаетесь в простом выводе текста на KOI-8, то попробуйте следующее:

1. Найти соответствующий KOI-8 шрифт для вашего принтера. Проверьте ftp архивы MS DOS - в Internete (например посмотрите на архив SimTel).
2. Прочитайте руководство и найдите в нем как загрузить такой шрифт в ваш принтер. Напишите простенькую программу, делающую это.
3. Запускайте эту программку из соответствующего `rc` файла при загрузке.

Таким образом, наличие символов Cyrillic в верхней части набора символов принтера позволит вам печатать тексты по-русски без дополнительных ухищрений.

Альтернативно к **KOI8-R** шрифтам вы можете попробовать использовать **Alt** шрифты. Для этого имеются две причины:

- Вероятно найти **Alt** шрифты намного проще, так как те были очень широко распространены во времена M$-ДОС.
- Наличие соответствующего **Alt** шрифта позволит вам печатать также и псевдо - графические символы.

Однако, в этом случае, вы должны будете преобразовать ваши тексты из **KOI8-R** в **Alt** перед посылкой их на принтер. Это не проблема, так как имеется множество программ, делающих это (для примера смотрите раздел Символьная перекодировка), так что вам нужно только вызвать такую программку из файла `/etc/printcap` в "`if` поле". Например, с программой `translit` можно сделать следующее:

```
if=/usr/bin/translit -t koi8-alt.rus
```

Для подробностей смотрите `printcap(5)`.

---

9. Локализация и Интернационализация

Пока, я описывал, как заставить различные программы понять кириллицу. Обычно, каждая программа требовала, чтобы это был ее собственный метод, как правило, чрезвычайно отличный от других. Кроме того, у некоторых программ была незавершенная поддержка языков отличных от английского. Не говоря уж об их неспособности взаимодействовать, используя родной язык пользователя вместо английского.

Проблемы, перечисленные выше сильно подавляют, так как программное обеспечение редко создается только для местного рынка. Переработка существенных частей программного обеспечения каждый раз при входе на новый международному рынок, очень неэффективна; и интернациональная поддержка, осуществляемая собственными средствами программы уникальным и присущим только ей способом, в терминах долгосрочного планирования так же не блестящая идея.

Следовательно, возникает потребность в стандартизации. И стандарт есть.

Все связанное с вышеперечисленными проблемами разделено в соответствии с двумя базисными концепциями: **localization** и **internationalization**. Под локализацией мы имеем в виду создание программ, способных обрабатывать различные языковые соглашения для различных стран. Позвольте привести пример. Формат даты выданный в Соединенных Штатах - имеет вид ММ/ДД/ГГ. Однако в России, наиболее популярный формат - ДД.ММ.ГГ. Другие проблемы включают в себя представление времени, форматы числа и представления валюты. Кроме этого, один из наиболее важных аспектов локализации - это определение соответствующих классов символов, то есть определение: какие символы в наборе символов являются "кирпичиками" языка (буквами) и как они упорядочиваются. С другой стороны, локализация не работает со шрифтами.

Интернационализация (или **i18n** для краткости), как предполагается, решает проблемы, связанные со способностью программы, взаимодействуют с пользователем на его родном языке.

Обе эти концепции должны быть стандартизованы, давая программистам непротиворечивый путь создания программ, работающих в национальной среде.

Хотя стандартизация еще в процессе, но много ее частей уже фактически являются стандартом; так что они могут использоваться без особых проблемы.

Я опишу общую схема создания программ использующих описанные выше возможности стандартным способом. Так как это заслуживает отдельного документа, я буду давать только очень общее описание и указатели на более полные источники.

9.1 Locale

Одно из основных понятий локализации - **locale**. Под locale подразумевается набор соглашений, специфических для отдельно взятого языка в отдельно взятой стране. В общем случае говорить, что locale определяется только страной неправильно. Например, в Канаде могут быть определены два locale- язык Канада / Английский и язык Канада / Французский. Более того, язык Канада / Английский - не является эквивалентом языку Великобритания / Английский или Американский / Английский,

точно так же Канада / Французский язык - не эквивалент языку Франция / Французский или языку Швейцария / Французский.

Locale с точки зрения пользователя

Каждая locale - это специальная база данных, определяющая по крайней мере следующие правила и соглашения:

1. Классификация символов и преобразований
2. Представление валюты
3. Представление чисел (то есть. Десятичные символы)
4. Формат даты / времени

В RedHat Linux (как вероятно и во многих других дистрибутивах Linux), имеются фактически две *базы* данных locale: одна для библиотеки C (`libc`), а другая для *X* библиотек. В идеальном случае должна иметься только одна база данных locale для всего.

Чтобы изменить значение locale по умолчанию, обычно достаточно установить системную переменную `LANG`. Например, как это делается в `sh`:

```
LANG=ru_SU
export LANG
```

Вы можете проверить действие этой команды сразу же, если запустите команду `date`. Результатом должен быть вывод дня, недели и месяца на русском языке.

RedHat 5.x определяет KOI8-R locale как `ru_SU`, по этой причине я и использую его. Более очевидное название `ru_RU` используется для locale основанного на `iso-8859-5` кодировки.

Иногда, вы можете захотеть изменить только один аспект locale без изменения других. Например, вы можете захотеть (Бог знает почему) пользоваться с `ru_SU` locale, но печатаемые числа должны будут соответствовать стандарту POSIX один. В подобных случаях, имеется набор системных переменных, которые Вы можете задать чтобы сконфигурировать соответствующие части locale. Например в нашем случае это бы выглядело так:

```
LANG=ru_SU
LC_NUMERIC=POSIX
export LANG LC_NUMERIC
```

Подробнее , см. `locale(7)`.

Теперь давайте держаться поближе к специфике Linux. К сожалению, в Linux `libc` версии 5.3.12, входящей в дистрибутив RedHat 4.1 отсутствует русская locale. В данном случае ее надо скачать из Interneta (я, однако, не знаю точного адреса).

Чтобы проверить, для каких языков у вас есть locale, выполните '`locale -a`'. Это выведет список всех locale из баз данных доступных libc.

Что касается библиотек X, то они имеют свою собственную базу данных locale. В версии которую я использую (XFree86 3.3), уже имеется российская база данных locale. Я не уверен есть ли она в предыдущей версии. В любом случае, вы можете проверить это, изучив директорию /usr/lib/X11/locale/ (в большинстве систем). В моем случае, уже есть подкаталоги, именованные koi8-r и даже iso8859-5..

Locale зависимое программирование

С locale программа не должна знать о различных символьных преобразованиях и правилах сравнения, описанных выше. Вместо этого, они используют специальный API, который действует по правилам, определенным locale. Кроме того, нет необходимости для программы, пользоваться только одной locale для соблюдения всех правил- возможно пользоваться другими правилами, описанных в других locale (хотя такой метод не очень хорош).

Из man setlocale(3):

> Программа может быть сделана переносимой для всех locale, вызывая setlocale(LC_ALL, "") после инициализации программы, используя значения, возвращенные из localeconv() запрос для locale - зависимой информации и используя strcoll() или strxfrm() для сравнения строк.

Довольно легко определить четыре уровня программной локализации:

1. **Чисто 8ми битное** программное обеспечение. То есть программа вызывает setlocale(). Она не делает каких-либо предположений относительно 8-ого бита каждого символа, использует пользовательские функции из ctype.h и ограничения из limits.h, и заботится относительно signed/unsigned результата. Очень важно, чтобы программа **не** делала каких-либо предположений относительно характера набора символов и их упорядочения. То есть следует воздержаться от следующих конструкций при программировании:

```
2.      if (c >= 'A' && c <= 'Z') {
3.          ...
```

   Взамен во всех таких случаях должны использоваться, макрокоманды из locale зависимого файла заголовка ctype.h.

4. Форматы, методы сортировки, размеры листа бумаги. Программа использует strcoll() и strxfrm() вместо strcmp() для строк, использует time(), localtime(), и strftime() для работы со временем, и в заключение, использует localeconv() для правильного представления чисел и валюты.
5. Видимый текст складывается в каталоги сообщений/. Программа должна локализовать весь видимый текст в специальных **каталогах сообщений**. Они содержат соответствия строк на английском и их переводы на другие языки. Выбор сообщений соответствующих языку окружения выполнен так, что полностью прозрачен и для программы и для пользователя. Чтобы использовать эти средства, программа должна вызвать gettext() (Sun/POSIX стандарт), или catgets() (X/Open стандарт). Подробнее см. раздел i18n .
6. EUC/Unicode поддержка. На этом уровне, программа не использует тип char. Взамен это она использует wchar_t, который определяет объекты, достаточно

большие, чтобы содержать символы Unicode. ANSI C определяет этот тип данных и соответствующий API.

Для выяснения подробностей, смотрите например (Voropay1) или (SingleUnix).

9.2 интернационализация

В то время как локализация описывает, как адаптировать программу к иностранному окружению, **интернационализация** (или **i18n** для краткости) детализирует способы общения программы с не-англоговорящим пользователем.

Прежде, это делалось с помощью создания абстракций сообщений, для вывода их из кода программы. Теперь, такой механизм (более или менее) стандартизирован. И, конечно, есть его free реализации!

Проект GNU наконец стал на путь создания интернационализированных прикладных программ. Ulrich Drepper (`drepper@ipd.info.uni-karlsruhe.de`) разработал пакет `gettext`. Этот пакет лежит во всех GNU архивах, например в prep.ai.mit.edu. Он позволяет вам разрабатывать программы в направлении, двигаясь в котором вы можете легко заставить их поддерживать большее количество языков. Я и не предполагаю описывать методы программирования, еще и потому, что `gettext` пакет поставляется с превосходным руководством.

Просьба о сотрудничестве: Если вы хотите изучить `gettext` пакет и сделать свой вклад в проект GNU или просто сделать вклад без всякого изучения, то вы можете сделать это! GNU становится международным, так что все утилиты делаются locale зависимыми. Проблема состоит в том, чтобы переводить сообщения от Английского языка на Русского (и другие языки, конечно если захотите). В общем, что следует сделать: вы должны получить специальный `.po` файл, состоящий из Английских сообщений для неких утилит, и связать каждое сообщение с его эквивалентом на русском. В конечном счете, это заставит говорить систему Русский, если пользователь захочет этого! Для для подробностей войдите в контакт с Ulrich Drepper (drepper@ipd.info.uni-karlsruhe.de).

---

10. Совместимость

Следовать стандарту это еще не все. В реальной жизни надо обеспечить еще и обратную совместимость. В нашем случае, это означает, что наши настройки не должны препятствовать созданию данных, с помощью других кодировок, отличных от стандартной. Это могут быть данные в **Alt (cp866)** или в **cp1251**. Также должна быть возможность запускать русскоязычные программы из MS-DOS.

В большинстве случаев (кроме HTTP), достаточно обеспечить конвертацию данных в **KOI8-R**. Если мы говорим о данных с простой структурной организацией, то это просто - смотрите раздел Символьная перекодировка.

Другое дело это данные с определенной структурой. В этом случае действия не так тривиальны. Я попробую описать стандартные подходы для решения этой проблемы.

10.1 MIME-based data compatibility

10.2 Совместимость данных в MIME формате

MIME формат - стандартное архитектурно независимое представление данных. Первоначально это представление данных было разработано для письменных сообщений, а сейчас его используют и в других местах. Стандарт MIME определяет формат, который открыт для расширений и позволяет поддержку и работу со специфическими данными. Например, если я послал письмо, содержащее **MIME объект** `video/mpeg` типа (MPEG файлы), моя программа приема электронных сообщений автоматически декодирует его и запустит MPEG проигрыватель.

Большинство UNIX программ, предлагающих MIME сервис, для этих целей используют пакет `metamail`, который содержит набор утилит и файлов данных для работы с объектами MIME. Несколько файлов конфигурации (`/etc/mailcap` - для системной настройки и `~/.mailcap` - для пользовательской настройки) определяют директивы для работы с объектами MIME различных типов.

Поэтому, если вы получили поток MIME данных, содержащий текст в одной из устаревших кодировок, вы можите определить соответствующие MIME-директивы для конвертации такого текста в KOI8.

Ниже перечислены MIME-законы, которые описывают правила работы с обычными текстами и текстами в richtext формате, использующих не особенно нужные (устаревшие) кодировки, описанные выше. Вы можете вставить эти директивы в один из файлов конфигурации MIME.

Обратите внимание: Эти директивы используют пакет `translit`, для того, чтобы производить само преобразование. Для более полной информации об этой программе и для информации по перекодировки смотрите раздел Символьная перекодировка.

```
text/plain; translit -t cp1251-koi8.rus < %s; test=test \
    "`echo %{charset} | tr '[A-Z]' '[a-z]'`"  = cp1251; copiousoutput

text/richtext; translit -t cp1251-koi8.rus < %s; test=test \
    "`echo %{charset} | tr '[A-Z]' '[a-z]'`"  = cp1251; copiousoutput

text/plain; translit -t alt-koi8.rus < %s; test=test \
    "`echo %{charset} | tr '[A-Z]' '[a-z]'`"  = cp866; copiousoutput

text/richtext; translit -t alt-koi8.rus < %s; test=test \
    "`echo %{charset} | tr '[A-Z]' '[a-z]'`"  = cp866; copiousoutput

text/plain; translit -t alt-koi8.rus < %s; test=test \
    "`echo %{charset} | tr '[A-Z]' '[a-z]'`"  = alt; copiousoutput

text/richtext; translit -t alt-koi8.rus < %s; test=test \
    "`echo %{charset} | tr '[A-Z]' '[a-z]'`"  = alt; copiousoutput
```

Достаточно, это работает только в случае обычного текст. Бинарные файлы данных должны сами отрабатывать данные в различных кодировках (По крайней мере, это должны делать программы, создавшие их). Поэтому если вы послали файл Microsoft

Word в кодировке **cp1251**, то с этим должна разобраться программа, в которой вы читаете этот текст (Например M$ Word или Applix Words).

К несчастью действительная ситуация далека от идеала. Много программных продуктов имеют собственные идеи по поводу того, как использовать MIME. До недавнего времени Microsoft Mail использовал испорченный механизм работы с MIME. Хотя и Netscape Navigator/Communicator клиент отправки/приема сообщений известен тем, что шлет текст письма в **cp1251**, а его заголовок в *charset=koi8-r* кодировки и наоборот.

10.3 Символьная перекодировка

В Internet можно найти множество программ, перекодирующих кириллические тексты. Каждая из них имеет свою изюминку и отличается степенью поддержки кириллицы.

С моей точки зрения утилиты должны быть стандартными. В нашем, частном, случае такой "стандартной" утилитой является `GNU recode`. К несчастью версия, которая "обитает" на официальном GNU сайте (3.4) пока не поддерживает кириллицу (только **ISO-8859-5**). Я сделал набор таблиц перекодировки для **KOI8-R**, **Alt**, и **cp1251** и послал их `recode` - координатору. Он обещал обеспечить поддержку кириллицы в следующем релизе программы. Как только это случится, то я перепишу этот параграф, для того чтобы рекомендовать `GNU recode`, как стандартную утилиту перекодировки для кириллицы.

Тем временем, я пока рекомендую пакет translit. Он поддерживает много популярных кодировок и даже способен создавать TeX файлы (см. раздел tex) на русском языке. Кроме того, для пользователей RedHat существует RPMка - RPM package.

Для других программ преобразования - загляните на страничку SovInformBureau или на ftp.funet.fi. Вы даже можете использовать специальную моду для `emacs` (Смотрите раздел Emacs и XEmacs).

10.4 Кириллические имена файлов в файловой системе M$ Windows

Windows имеет возможность давать имена файлов в кодировке Unicode, поэтому пользователи могут давать осмысленные имена своим файлам. Linux, однако не дает полной поддержки для Unicode, поэтому если вы монтируете диск Windows *VFAT*, то вы увидите что подобные имена состоят из вопросительных знаков и прочего мусора.

Эдесь указан путь как получить правильное преобразование:

1. Убедитесь, что ваше ядро скомпилирована с поддержкой codepage (в частности с поддержкой кодовых страниц 866 и KOI8-R)
2. Добавьте следующие опции при выполнении mount команды `mount`:
3. `codepage=866,iocharset=koi8-r`

Для подробностей смотрите `Documentation/filesystems/vfat.txt` в исходниках ядра Linux.

10.5 Поддержка кириллицы в DOS эмуляторе

Это, видимо, единственный программный продукт, которые требует присутствия `Alt` кодировки. Основанием является то, что `Alt-` это родная кириллическая кодовая страница DOS. Большенство программ, работающих в DOS с кириллицей ориентированы на `Alt`.

Для консольной версии (`dos`) вы должны загрузить только клавиатуру и экранный драйвер. Большинство драйверов DOS будет прекрасно работать. Я лично использую `rk` драйвер А. Страхова, который работает и для консоли, и X версии `dosemu`. Другая альтернатива это `r` драйвер Вадима Курлянда. Он прекрасно настраивается и поддерживает много кодировок в том числе, `Alt` и `KOI8`. Однако он не будет работать для X Windows (по крайней мере версия 1.14, которую я использую, (Примечание переводчика: сейчас есть версия 2.0 этого драйвера и его развитие прекращено).

Оба русификатора можно найти на большинстве Российских Internet архивах, например на ftp архиве Курчатовского Института Ядерной Физики.

Для Ховых версии `dosemu` вы должны установить соответствующий X шрифт. Алексей Богданов прислал мне такой шрифт по электронной почте. Это - родной шрифт `vga` из дистрибутива `dosemu`, измененный для `Alt` кодировки. К сожалению я не знаю, кто автор этого шрифта и где его официальная страничка. Я помещу этот шрифт в мой каталог на авторской страничке

Для установки шрифтов для `dosemu` вы должны To setup the font for `dosemu` you should

- Сделайте этот шрифт доступным X. серверу, как это сделать описано в Настройка шрифтов X Windows.
- Прикрутите этот шрифт к `dosemu`. Если шрифт только заменяет первоначальный шрифт `vga`, то он будет опознан по умолчанию. Иначе, вы должны описать его в `/etc/dosemu.conf`::
- `# Font to use (without filename extensions). For example:`
- `X { updatefreq 8 title "MS DOS" icon_name "xdos" font "vga-alt"}`

И в заключение, вы должны загрузить драйвер клавиатуры. Обратите внимание, вам не нужны экранные драйверы в X окне. Не все драйверы будут работать, но по крайней мере два из них будут: `rk` А. Страхова, и `cyrkeyb` Pete Kvitek.
