The Linux Cyrillic HOWTO

Alexander L. Belikoff, (abel@bfr.co.il) v3.0, 13 August 1997

This document describes how to set up your Linux box to typeset, view and print the documents in the Russian language.

Contents

1 General notes **2**

1.1 Introduction . . . 2

1.2 Availability and feedback . . . 3

1.3 Acknowledgments and copyrights . . . 3

2 Characters and codesets **4**

3 Text mode setup **6**

3.1 Linux Console . . . 7

3.2 FreeBSD Console . . . 8

4 The X Window System **9**

4.1 The X fonts. . . . 9

4.2 The input translation . . . 11

4.2.1 The table of characters . . . 12

4.2.2 The mode switching rules . . . 14

5 Cyrillic support in TeX and LaTeX **17**

5.1 Using the Washington Cyrillic . . . 17

5.2 KOI-8 package for teTeX . . . 18

5.3 Using the cmcyralt package for LaTeX . . . 18

5.4 Using the CyrTUG package . . . 19

6 Cyrillic in PostScript **19**

6.1 Adding Cyrillic fonts to Ghostscript . . . 20

7 Print setup **20**

7.1 Pre-loading Cyrillic fonts into a non-PostScript printer . . . 21

7.2 Printing with different fonts . . . 21
7.3 Converting text to TeX . . . 22
7.4 Text to PostScript converters . . . 22

8 Miscellaneous utilities setup 23
8.1 bash . . . 23
8.2 csh/tcsh . . . 23
8.3 emacs . . . 23
8.4 ispell . . . 25

9 joe 25
9.1 ksh . . . 25
9.2 less . . . 25
9.3 lynx . . . 25
9.4 mc (The Midnight Commander) . . . 25
9.5 Netscape navigator . . . 25
9.5.1 Basic setup . . . 26
9.5.2 Advanced setup . . . 26
9.6 pine . . . 26
9.7 rlogin . . . 26
9.8 sendmail (aka "The Doom of a Sysadmin") . . . 26
9.9 zsh . . . 26

10 Useful Tools 27
10.1 Conversion Utilities . . . 27
10.2 Programmer's tools . . . 27

11 Summary of the various useful resources 27

1 General notes

1.1 Introduction

This document covers the things you need to successfully typeset, view, and print documents in Russian under Linux. Although this document assumes your using Linux as an operating system, most of

information presented is equally applicable to many other Unix flavors. I shall try to keep the distinction as visible as possible.

There are a number of popular Linux distributions. As an example system I describe the RedHat 3.0.3 Linux (Picasso) and the RedHat 4.1 Linux (Vanderbildt) - the one I am personally using. Nevertheless, I shall try to highlight the differences, if they exist, in the Slackware Linux setup.

Since such setup directly modifies and extends the Operating System, you should understand, what you are doing. Even though I tried to keep things as easy as possible, having some experience with a given piece of software is an advantage. I am not going to describe what the X Window System is or how to typeset the documents with TeX and LaTeX, or how to install printer in Linux. Those issues are covered in other documents.

For the same reason, in most cases I describe a system-wide setup, by default requiring *root* privileges. Still, if there is a possibility for user-level setup, I'll try to mention it.

NOTE: The X Window System, TeX and other Linux components are complex systems with a sofisticated configuration. If you do something wrong, you can not only fail with Russian setup, but to break the component as well, if not the entire system. This is not to scare you off, but merely to make you understand the seriousness of the process and be careful. Preliminary backup of the config files is **highly** recommended. Having a guru around is also advantageous.

1.2 Availability and feedback

This document is available at sunsite.unc.edu or tsx-11.mit.edu as a part of the *Linux Document Project*. Also, it may be available at various FTP sites containing Linux. Moreover, it may be included as a part of Linux distribution.

You may also get it directly from the author at ftp.netvision.net.il.

If you have any suggestions or corrections regarding this document, please, don't hesitate to contact me as belikoff@netvision.net.il. Any new and useful information about Cyrillic support in various Unices is *highly appreciated*. Remember, it will help the others.

1.3 Acknowledgments and copyrights

Many people helped me (and not only me) with valuable information and suggestions. Even more people contributed software to the public community. I am sorry if I forgot to mention somebody.

So, here they go:

- Bas V. de Bakker
- David Daves
- Serge Vakulenko
- Sergei O. Naoumov
- Winfried Truemper

- Ilya K. Orehov
- Michael Van Canneyt

This document is Copyright (C) 1995,1997 by Alexander L. Belikoff. It may be used and distributed under the usual Linux HOWTO terms described below.

The following is a Linux HOWTO copyright notice:

> *Unless otherwise stated, Linux HOWTO documents are copyrighted by their respective authors. Linux HOWTO documents may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies. Commercial redistribution is allowed and encouraged; however, the author would like to be notified of any such distributions.*
>
> *All translations, derivative works, or aggregate works incorporating any Linux HOWTO documents must be covered under this copyright notice. That is, you may not produce a derivative work from a HOWTO and impose additional restrictions on its distribution. Exceptions to these rules may be granted under certain conditions; please contact the Linux HOWTO coordinator at the address given below.*
>
> *In short, we wish to promote dissemination of this information through as many channels as possible. However, we do wish to retain copyright on the HOWTO documents, and would like to be notified of any plans to redistribute the HOWTOs.*

If you have questions, please contact Greg Hankins, the Linux HOWTO coordinator, at gregh@sunsite.unc.edu. You may finger this address for phone number and additional contact information.

Unix is a technology trademark of the X/Open Ltd.; MS-DOS, Windows, Windows 95, and Windows NT are trademarks of the Microsoft Corp.; The X Window System is a trademark of The X Consortium Inc. Other trademarks belong to the appropriate holders.

2 Characters and codesets

In order to understand and print characters of various languages, the system and software should be able to distinguish them from other characters. That is, each unique character must have a unique representation inside the operating system, or the particular software package. Such collection of all unique characters, that the system is able to represent at once, is called a *codeset*.

At the time of the most operating system's creation, nobody cared about software being multilingual. Therefore, the most popular codeset was (and actually is) an **ASCII** (American Standard Code for Information Interchange).

The *standard ASCII* (aka 7-bit ASCII) comprises 128 unique codes. Some of them ASCII defines as real printable characters, and some are so-called *control characters*, which had special meanings in the old communication protocols. Each element of the set is identified by an integer *character code* (0-127). The

subset of printable characters represents those found on the typewriter's keyboard with some minor additions. Each character occupies 7 least significant bits of a byte, whereas the most significant one was used for control purposes (say, transmission control in old communication packages).

The 7-bit ASCII concept was extended by 8-bit ASCII (aka *extended ASCII*). In this codeset, the characters' codes' range is 0-255. The lower half (0-127) is pure ASCII, whereas the upper one contains 127 more characters. Since this codeset is backward compatible with the ASCII (character still occupies 8 bit, the codes correspond the old ASCII), this codeset gained wide popularity.

The 8-bit ASCII doesn't define the contents of the upper half of the codeset. Therefore the ISO organization took the responsibility of defining a family of standards known as *ISO 8859-X* family. It is a collection of 8-bit codesets, where the lower half of each codeset (characters with codes 0-127) matches the ASCII and the upper parts define characters for various languages. For example, the following codesets are defined:

- `8859-1` - Europe, Latin America (also known as *Latin 1*)
- `8859-2` - Eastern Europe
- `8859-5` - Cyrillic
- `8859-8` - Hebrew

In Latin 1, the upper half of the table defines various characters which are not part of the English alphabet, but are present in various european languages (german umlauts, french accentes etc).

Another popular extended ASCII implementation is so-called *IBM codepage* (named after some computer company, that developed this codeset for it's infamous personal computers). This one contains pseudo-graphic characters in the upper half.

Software, that doesn't make any assumptions about the 8-th bit of the ASCII data is called *8-bit clean*. Some older programs, designed with 7-bit ASCII in mind are not 8-bit clean and may work incorrectly with your extended ASCII data. Most of packages, however, are able to deal with the extended ASCII by default, or require some very basic setup. **NOTE:** before posting the question *"I did all setup right, but I cannot enter/view Cyrillic characters!"*, please consult the section 8 for the notes on the program, you are using.

For information about making your software 8-bit clean, see section 10.2.

Since on most systems character occupies 8 bits, there is no way to extend ASCII more and more. The way to implement new symbols in ASCII-based codesets is creation of other extended ASCII implementations. This is the way, the Cyrillic ASCII set is implemented.

We already mentioned *ISO 8859-5* standard as the one defining the Cyrillic codeset. But as it often happens to the standards, this one was developed without taking into account the real practices in the former USSR. Therefore, one thing that standard really achieved was another degree of confusion. I wouldn't say that *ISO 8859-5* is widely used anywhere.

Other standards for Cyrillic include the so-called *Alt* codeset and *Microsoft CP1251* codepage. The former one was developed by (who?) for MS-DOS quite a while ago. Back then, there was not very buzz yet about

internetworking, so the intention was to make it as compatible as possible with the IBM standard. Therefore the Alt codeset is effectively the same IBM codepage, where all specific European characters in the upper half were replaced with the Cyrillic ones, leaving the pseudographic ones. Therefore, it didn't screw the text windowing facilities and provided Cyrillic characters as well. The *Alt* standard is still alive and extremely popular in MS-DOS.

Microsoft CP1251 codepage is just an attempt of Microsoft to come up with the new standard for Cyrillic codeset in Windows. As far as I know, it is not compatible with anything else (not very surprizing, huh?)

And finally there is *KOI-8*. This one is also quite old, but it was designed wisely and nowadays the design points of it look really useful.

Again, it is compatible with ASCII, and the Cyrillic characters are located in the upper half. But the main design point of *KOI-8* is that the Cyrillic characters' positions must correspond to the English characters with the same phonetics. Namely, if we set the eighth bit of the English character `'a'`, we'll get the Cyrillic `'a'`. This means that, given the Cyrillic text written in KOI-8, we can strip the eighth bit of each character *and we still get a readable text, although written with English characters!* This is very important now, since there are many mailers on the Internet, that just strip the eighth bit silently, being sure that every single soul on the face of the Earth speaks English.

Not surprisingly, *KOI-8* quickly became a de-facto standard for Cyrillic on the Internet. Andrew A. Chernov did a tremendous amount of work to make a standard in this area. He is an author of RFC 1489 (*"Registration of a Cyrillic Character Set"*).

These two standards differ only in positions of the cyrillic characters in the table (that is in cyrillic character codes).

The principal difference is that the Alt codeset is used by MS-DOS users only, whereas KOI-8 is used in Unix, as well as in MS-DOS (though in the latter KOI-8 is much less popular). Since we are doing the right thing (namely working in the Unix operating system), we shall focuse mostly on KOI-8.

As for the ISO standard, it is more popular in Europe and the US as a standard for Cyrillic. The leader in Russia is definitely KOI-8.

There are other standards, which are different from ASCII and much more flexible. **Unicode** is most known. However, they are not implemented as good as the basic ones in Unix in general and Linux in particular. Therefore, I am not describing them here.

3 Text mode setup

Generally, the text mode setup is the easiest way to show and input Cyrillic characters. There is one significant complication, however: the text mode fonts and keyboard layout manipulations depend on terminal driver implementation. Therefore, there is no portable way to achieve the goal across different systems.

Right now, I describe the way to deal with the Linux console driver. Thus, if you have another system, don't expect it to work for you. Instead, consult your terminal driver manual. Nevertheless, send me any information you find, so I'll be able to include it in further versions of this document.

3.1 Linux Console

The Linux console driver is quite a flexible piece of software. It is capable of changing fonts as well as keyboard layouts. To achieve it, you'll need the kbd package. Both RedHat and Slackware install kbd as part of a system.

The kbd package contains keyboard control utilities as well as a big collection of fonts and keyboard layouts.

Cyrillic setup with **kbd** usually involves two things:

1. Screen font setup. This is performed by the `setfont` program. The fonts files are located in `/usr/lib/kbd/consolefonts`.

   NOTE: Never run the `setfont` program under X because it will hang your system. This is because it works with low-level video card calls which X doesn't like.

2. Load the appropriate keyboard layout with the `loadkeys` program.

NOTE: In RedHat 3.0.3, `/usr/bin/loadkeys` has too restrictive access permissions, namely 700 (`rwx------`). There are no reasons for that, since everyone may compile his own copy and execute it (the appropriate system calls are not root-only). Thus, just ask your sysadmin to set more reasonable permissions for it (for example, 755).

The following is an excerpt from my `cyrload` script, which sets up the Cyrillic mode for Linux console:

```
if [ notset.$DISPLAY != notset. ]; then
    echo "`basename $0`:  cannot run under X"
    exit
fi

loadkeys /usr/lib/kbd/keytables/ru.map
setfont /usr/lib/kbd/consolefonts/Cyr_a8x16
mapscrn /usr/lib/kbd/consoletrans/koi2alt
echo -ne "\033(K"              # the magic sequence
echo "Use the right Ctrl key to switch the mode..."
```

Let me explain it a bit. You load the appropriate keyboard mapping. Then you load a font corresponding to the *Alt* codeset. Then, in order to be able to display text in *KOI8-R* correctly, you load a *screen translation table*. What it does is a translation of *some* characters from the upper half of the codeset to the *Alt* encoding. The word 'some' is crucial here - not all characters get translated, therefore some of them, like IBM pseudographic characters get unmodified to the screen and display correctly, since they are compatible with the *Alt* codeset, as opposed to *KOI8-R*. To ensure this, run **mc** and pretend you are back to MS-DOS 3.3...

Finally, the magic sequence is important but I have no idea what on the Earth it does. I stole/borrowed/learned it from German HOWTO back in 1994, when it was like the only national language oriented HOWTO. *If you have any idea about this magic sequence, please tell me.*

Finally, for those purists, who don't wont to give the *Alt* codeset a chance, I'm attaching yet another version of the script above, using native *KOI8-R* fonts.

```
if [ notset.$DISPLAY != notset. ]; then
    echo "`basename $0`:  cannot run under X"
    exit
fi

loadkeys /usr/lib/kbd/keytables/ru.map
setfont /usr/lib/kbd/consolefonts/koi-8x16
echo "Use the right Ctrl key to switch the mode..."
```

However, don't expect nice borders in your text mode-based windowing applications.

Now you probably want to test it. Do the appropriate bash or tcsh setup, rerun it, then press the right `Control` key and make sure you are getting the cyrillic characters right. The '`q`' key must produce russian "`short i`" character, '`w`' generates "`ts`", etc.

If you've screwed something up, the very best thing to do is to reset to the original (that is, US) settings. Execute the following commands:

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

NOTE: unfortunately enough, the console driver is not able to preserve it's state (at least easily enough), while running the X Window System. Therefore, after you leave the X (or switch from it to a console), you have to reload the console russian font.

3.2 FreeBSD Console

I am not using FreeBSD so I couldn't test the following information. All data in this section should be treated as just pointers to begin with. The FreeBSD project homepage may have some information on the subject. Another good source is the relcom.fido.ru.unix newsgroup. Also, check the resources listed in section 11.

Anyway, this is what Ilya K. Orehov suggests to do in order to make FreeBSD console speak Russian:

1. In `/etc/sysconfig` add:

```
keymap=ru.koi8-r
keyrate=fast
# NOTE: '^[' below is a single control character
keychange="61 ^[[K"
cursor=destructive
```

```
scrnmap=koi8-r2cp866
font8x16=cp866b-8x16
font8x14=cp866-8x14
font8x8=cp866-8x8
```

2. In `/etc/csh.login`:

```
setenv ENABLE_STARTUP_LOCALE
setenv LANG ru_SU.KOI8-R
setenv LESSCHARSET latin1
```

3. Make analogous changes in `/etc/profile`

4 The X Window System

Like the console mode, the X environment also requires some setup. This involves setting up the input mode and the X fonts. Both are being discussed below.

4.1 The X fonts.

First of all, you have to obtain the fonts having the Cyrillic glyphs at the appropriate positions.

If you are using the most recent X (or XFree86) distribution, chances are, that you already have such fonts. In the late 1995, the X Window System incorporated a set of Cyrillic fonts, created by Cronyx. Ask your system administrator, or, if *you* are the one, check your system, namely:

1. Run '`xlsfonts | grep koi8`'. If there are fonts listed, your X server is already aware about the fonts.

2. Otherwise, run

   ```
   find -name crox\*.pcf\*
   ```

   to find the location of the Cyrillic fonts in the system. You'll have to `enable` those fonts to the X server, as I explain below.

If you haven't found such fonts installed, you'll have to do it yourself.

There is some ambiguity with the fonts. XFree86 docs claim that the russian fonts collection included in the distribution is developed by Cronyx. Nevertheless, you may find another set of Cronyx Cyrillic fonts on the net (eg. on ftp.kiae.su), known as the **xrus** package (don't confuse it with the `xrus` program, which is used to setup a Cyrillic keyboard layout. Hopefully, tha letter one was renamed to **xruskb** recently). **Xrus** has fewer fonts than the collection in Xfree86 (38 vs 68), but the latter one didn't go along with my 9.5 setup - it gave me some really huge font in the menubar. The **xrus** package doesn't have this problem.

I would suggest you to download and try both of them. Pick up the one which you'll like more. Also, I'm going to creat RPM packages soon for both collections and download them both to ftp.redhat.com and to my FTP site.

There are also older stuff, for example the **vakufonts** package, created by Serge Vakulenko, which was the base for the one in the X distribution. There are also a number of others. The important point is that the fonts' names in the old collection were not strictly conforming to the standard. The latter is fine in general, but sometimes it may cause various weird errors. For example, I had a bad experience with Maple V for Linux, which crashed mysteriously with the **vakufonts** package, but ran smoothly with the "standard" ones.

So, let's start with the fonts:

1. Download the appropriate fonts collection. The package for XFree86 may be found at any FTP site, containing the X distribution, for example, directly from the XFree86 FTP site. The **xrus** package may be found on ftp.kiae.su

2. Now when you have the fonts, you create some directory for them. It is generally a bad idea to put new fonts to the already existing font directory. So, place them, to, say, `/usr/lib/X11/fonts/cyrillic` for a system-wide setup, or just create a private directory for personal use.

3. If the new fonts are in BDF format (`*.bdf` files), you have to compile them. For each font do:

   ```
   bdftopcf -o <font>.pcf <font>.bdf
   ```

   If your server supports compressed fonts, do it, using the *compress* program:

   ```
   compress *.pcf
   ```

   Also, if you do want to put the new fonts to an already existing font directory. you have to concatenate the old and the new files named `fonts.alias` in the case both of them exist.

4. Each font directory in the X must contain a list of fonts in it. This list is stored in the file `fonts.dir`. You don't have to create this list manually. Instead, do:

   ```
   cd <new font directory>
   mkfontdir .
   ```

5. Now you have to make this font directory known to the X server. Here, you have a number of options:

   - System-wide setup for XFree86. If you are running this version of X, then append the new directory to the list of directories in the file `XF86Config`. To find the location of this file, see output of `startx`. Also, see **XF86Config(4/5)** for details.
   - System-wide setup through `xinit`. Add the new directory to the `xinit` startup file. See **xinit(1x)** and the next option for details.

- Personal setup. You have a special start-up file for the X - `~/.xinitrc` (or `~/.Xclients`, or `~/.xsession` for the RedHat users). Add the following commands to it:

```
xset +fp <new font directory>
xset fp rehash
```

6. Now restart your X. If you have done everything right, the tests in the beginning of the section will be successful. Also, play with **xfontsel(1x)** to make sure you are able to select the cyrillic fonts.

In order to make the X clients use the Cyrillic fonts, you have to set up the appropriate X resources. For example, I make the russian font the default one in my `~/.Xdefaults`:

```
*font:          6x13
```

Since my cyrillic fonts are first in the font path (see output of '`xset q`'), the font above is taken from the "cyrillic" directory.

This just a simple case. If you want to set the appropriate part of the X client to a cyrillic font, you have to figure out the name of the resource (eg. using **editres(1x)**) and to specify it either in the resource database, or in the command line. Here go some examples:

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

...will run xterm with some ugly font; and

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

...will set a Cyrillic Times font for the **Quit** button in `xfontsel`.

4.2 The input translation

In the newest X releases (X11R61 and higher) there are two "standard" input methods: the original one, working through the **xmodmap** utility, and the new one called *Xkb* (X KeyBoard). The very first thing you have to do is **to disable the Xkb method!** Don't get charmed by it's ability to set up a "russian keyboard". It looks like this method is using the Cyrillic keysyms defined in `keysymdef.h`. This file defines keysyms for many languages. The only problem is that those definitions have nothing to do with the extended ASCII codeset - the one most programs are only able to operate with! I hardly know any programs being able to grok the `keysymdef.h` keysyms, different from 8-bit ASCII. However our goal is to get the KOI8-R support to work.

To disable the `Xkb` support, browse through the `Keyboard` section of your `XF86Config` file and comment all lines starting with *Xkb* (case doesn't matter). Instead, put the following line:

```
XkbDisable
```

The `xmodmap` program.allows customization of codes emitted by various characters and their combinations. It sets the things up based on the file containing the translation table.

If you don't want to deal with all these tricks and you prefer having a solution right away, either download an appropriate xmodmap table, available at many sites dealing with the Cyrillic, for example, ftp.kiae.su or ftp.funet.fi. Also, I made the table, described below, available at my FTP site.

More convenient alternative is to install the xruskb package, which allows you to configure most of the input translation parameters without having to know about `xmodmap`.

The following is a simplified description of input customization. If you want to do more sophisticated tricks, refer to the **xmodmap(1)** or, even better, wait for the next major X release, which will hopefully address the current input problems.

In our case, the translation table should define two things:

- the character codes emitted by the alphanumeric characters, and
- the mode switching rules

4.2.1 The table of characters

This is basically a sequence of directives which assign the certain keysyms to a specified keycodes. The general syntax is the following:

`keycode` *code* `=` *sym1 sym2 sym3 sym4*

where *code* is the numerical code of the given key on the keyboard (refer to the standard table for your system. In my case it is stored in the file `/usr/lib/X11/etc/xmodmap.std`). The *sym*s define the keysyms emitted by that key in different conditions. *Sym1* is the keysym emitted by the key in a regular state, *sym2* corresponds the key in *shifted* state (usually when `Shift` is held down). *Sym3* and *sym4* define the keysyms emitted when the `Mode_switch` is active for the normal and shifted states respectively (*group 2*, according to the X Protocol Specification). In our case, the active `Mode_switch` corresponds to the Cyrillic input mode.

These should be either hexadecimal codes or the symbolic constants from `/usr/include/X11/keysymdef.h` (without leading "XK_").

Thus, if we wanted the key corresponding to the Latin 'a' generate the Russian 'a' in the alternative mode, we would write the following:

```
keycode 38  =  a  A  0xC1 0xE1
```

The reader might be curious why I don't use the `Cyrillic_a` and `Cyrillic_A` constants respectively. The answer is that it didn't work for me. I am not very familiar with the guts of the X Window System specification, but I've got the following explanation. The symbolic constants above have the values `0x6C1` and `0x6E1` respectively. This means that in really multi-lingual environment they could be successfully used without overlapping with any other character set. However the KOI-8 standard is not well suited for

such environment. Thus, since we want to retain compatible with the past, we will violate the rules of multi-lingual support in the X Window System.

The following is a table for the most popular russian JCUKEN keyboard layout (these tables are derived from the ones in the `vakufonts` package):

```
keysym  4            = 4              dollar         4       quotedbl
keysym  5            = 5              percent        5       colon
keysym  6            = 6              asciicircum    6       comma
keysym  7            = 7              ampersand      7       period
keysym  q            = q              Q              0xCA    0xEA
keysym  w            = w              W              0xC3    0xE3
keysym  e            = e              E              0xD5    0xF5
keysym  r            = r              R              0xCB    0xEB
keysym  t            = t              T              0xC5    0xE5
keysym  y            = y              Y              0xCE    0xEE
keysym  u            = u              U              0xC7    0xE7
keysym  i            = i              I              0xDB    0xFB
keysym  o            = o              O              0xDD    0xFD
keysym  p            = p              P              0xDA    0xFA
keysym  bracketleft  = bracketleft    braceleft      0xC8    0xE8
keysym  bracketright = bracketright   braceright     0xDF    0xFF
keysym  a            = a              A              0xC6    0xE6
keysym  s            = s              S              0xD9    0xF9
keysym  d            = d              D              0xD7    0xF7
keysym  f            = f              F              0xC1    0xE1
keysym  g            = g              G              0xD0    0xF0
keysym  h            = h              H              0xD2    0xF2
keysym  j            = j              J              0xCF    0xEF
keysym  k            = k              K              0xCC    0xEC
keysym  l            = l              L              0xC4    0xE4
keysym  semicolon    = semicolon      colon          0xD6    0xF6
keysym  apostrophe   = apostrophe     quotedbl       0xDC    0xFC
keysym  grave        = grave          asciitilde     0xA3    0xB3
keysym  z            = z              Z              0xD1    0xF1
keysym  x            = x              X              0xDE    0xFE
keysym  c            = c              C              0xD3    0xF3
keysym  v            = v              V              0xCD    0xED
keysym  b            = b              B              0xC9    0xE9
keysym  n            = n              N              0xD4    0xF4
keysym  m            = m              M              0xD8    0xF8
keysym  comma        = comma          less           0xC2    0xE2
keysym  period       = period         greater        0xC0    0xE0
```

Also, for those using the russian YAWERTY layout, I include the following table:

```
keysym  q             = q              Q              0xD1    0xF1
keysym  w             = w              W              0xD7    0xF7
keysym  e             = e              E              0xC5    0xE5
keysym  r             = r              R              0xD2    0xF2
keysym  t             = t              T              0xD4    0xF4
keysym  y             = y              Y              0xD9    0xF9
keysym  u             = u              U              0xD5    0xF5
keysym  i             = i              I              0xC9    0xE9
keysym  o             = o              O              0xCF    0xEF
keysym  p             = p              P              0xD0    0xF0
keysym  bracketleft   = bracketleft    braceleft      0xDB    0xFB
keysym  bracketright  = bracketright   braceright     0xDD    0xFD
keysym  a             = a              A              0xC1    0xE1
keysym  s             = s              S              0xD3    0xF3
keysym  d             = d              D              0xC4    0xE4
keysym  f             = f              F              0xC6    0xE6
keysym  g             = g              G              0xC7    0xE7
keysym  h             = h              H              0xC8    0xE8
keysym  j             = j              J              0xCA    0xEA
keysym  k             = k              K              0xCB    0xEB
keysym  l             = l              L              0xCC    0xEC
keysym  z             = z              Z              0xDA    0xFA
keysym  x             = x              X              0xD8    0xF8
keysym  c             = c              C              0xC3    0xE3
keysym  v             = v              V              0xD6    0xF6
keysym  b             = b              B              0xC2    0xE2
keysym  n             = n              N              0xCE    0xEE
keysym  m             = m              M              0xCD    0xED
keysym  backslash     = backslash      bar            0xDC    0xFC
keysym  grave         = grave          asciitilde     0xC0    0xE0
keysym  equal         = equal          plus           0xDE    0xFE
keysym  3             = 3              numbersign     3       0xDF
keysym  4             = 4              dollar         4       0xFF
```

4.2.2 The mode switching rules

This is the trickiest part of the X Cyrillic setup. You should define the conditions in which the current mode is switched between the regular and the Cyrillic one. Thanks to the wise X input translation design, it is non-portable, inflexible, incomplete, and, generally speaking, it **sucks**. No surprise, people switch to MS Windows.

Currently, there is a lot of activity in development of XFree86 in general and X input model in particular. The result is, you configuration must work with one XFree version and not work with a different one. Even worse, sometimes things differ accross different servers in the same distribution.

First of all, I'll describe my setup, which works for me *now*. I am running XFree86 3.3 SVGA server. In my XF86Config file (in the RedHat distribution it is located in `/etc/X11`) I have the following directives in the "Keyboard" section:

```
LeftAlt         Meta
RightAlt        ModeShift
ScrollLock      ModeLock
RightCtl        Control
```

NOTE: The way to set up the mode switching rules via `XF86Config` file is XFree86-specific. It won't work for other X versions.

Now, I load the table described in the last section and stored in some file using the **xmodmap(1X)** utility:

```
xmodmap  <table_file>
```

Now I am able to shift to the Cyrillic input mode using right `Alt` key and to switch to Cyrillic and back using `ScrollLock`. You'll be surprised to learn that neither `CapsLock` nor `Control` keys work in the Cyrillic input mode. Well, say it again: the X input model **sucks**!

The rest of this section is a concise description of generic X input translation configuration. It used to work for older XFree86 releases but exposes various problems now. I left it here, so that the most curious of you would be able to try some of this stuff and make use of it.

What you should do is:

- Assign the `Mode_switch` keysym to some key, and
- Add `Mode_switch` to some spare modifier map

Now the key to which the `ModeShift` is assigned will act as a mode switch. This means that while it is held down, the keyboard is in alternative mode.

Moreover, if you add a *lockable* key to that modifier's map, this key will lock the alternative mode.

Note: There are some problems however. Serge Vakulenko (`vak@cronyx.com`) pointed out that the different X Server implementations may have different rules of assignments the mode switches (like, for example, some servers restrict the set of the keys which may work in toggle mode to, say, `CapsLock`, `NumLock`, and `ScrollLock`). Hopefully, this is a subject to change in the next release of the X Window System. For more details, see the X Protocol specification.

Unfortunately, I didn't manage to make the `CapsLock` key have the same functionality in the alternative mode, namely, to lock the upper case. It seems to me, it is impossible to do it, because of the idiotic X input translation design. If I am wrong, please correct me.

Let's see an example. Suppose, one wants to use the right `Alt` as a mode switch and the `ScrollLock` as as a mode lock. First of all, one should check the default modifiers' map. This is accomplished by running the `xmodmap` without arguments:

```
$ xmodmap

xmodmap:  up to 2 keys per modifier, (keycodes in parentheses):

shift       Shift_L (0x32),  Shift_R (0x3e)
lock        Caps_Lock (0x42)
control     Control_L (0x25)
mod1        Alt_L (0x40),  Alt_R (0x71)
mod2        Num_Lock (0x4d)
mod3
mod4
mod5
```

According to the above, the plan of attack is the following:

1. remove the `Alt_R` key from the `mod1` map
2. assign the `Mode_switch` keysym to the `Alt_R` key
3. assign the `Scroll_Lock` keysym to the keycode 78 (the code of the *actual* ScrollLock)
4. add the `Mode_switch` to the spare (`mod3`) map, and
5. add the `Scroll_Lock` keysym to the `mod3` map

Thus, here is the solution:

```
remove mod1 = Alt_R
keysym Alt_R = Mode_switch
keycode 78 = Scroll_Lock
add mod3 = Mode_switch
add mod3 = Scroll_Lock
```

If you use the latter solution, you may combine both the table and the mode directives in your `~/.Xmodmap` file.

Such files are generally supplied with the various X Cyrillic stuff packages. The good example is the tables in the old package by Serge Vakulenko described above.

Once you have such file containing the table, you should run the command:

```
xmodmap filename
```

...every time you start X. Modify your X startup file to do it. The system-wide file is `/usr/lib/X11/xinit/xinitrc`; the personal one is either `~/.xinitrc`, or `~/.Xclients`, or `~/.xsession`, depending on what you have.

NOTE: If `xmodmap` complains on your table, try to load the default table first. The default one is usually located in `/usr/lib/X11/etc/xmodmap.std`.

5 Cyrillic support in TeX and LaTeX

In this section I'll describe several ways to make TeX and LaTeX typeset Cyrillic texts. There are several ways, which differ in setup sophistication and usage convenience. For example, one possibility is to start without any preliminary setup and use the *Washington AMSTeX Cyrillic fonts*. On the other hand, you may install a LaTeX package, providing a very high degree of Cyrillic setup. I have an experience with two such packages. One is the `cmcyralt` package by Vadim V. Zhytnikov (`vvzhy@phy.ncu.edu.tw`) and Alexander Harin (`harin@lourie.und.ac.za`), and the other one is the `LH` package by the *CyrTUG* group with styles and hyphenation for LaTeX2e by Sergei O. Naoumov (`serge@astro.unc.edu`). I'll describe both.

Note, that there are two versions of LaTeX available - 2.09 is the old one, while 2e is a new pre-3.0 release. If you are using LaTeX 2.09, then switch quickly to the 2e. The latter retains compatibility with the old one, but has much more features. Hopefully, version 3 will be released soon. I describe a LaTeX 2e setup.

Also, both of these packages require the Cyrillic text to be typeset using the *Alt* codeset, not *KOI-8*! This is caused by historical reasons, since the creators of these packages used to work with `EmTeX` - the MS-DOG version of TeX (they didn't know about Linux yet :-). Switching to the *KOI-8* requires some effort and is being expected to be done soon. So far, use some utility to convert your russian text from *KOI-8* to *Alt*. See section 10.1.

5.1 Using the Washington Cyrillic

This package was created for the American Mathematic Society to provide documents with Russian references. Therefore, the authors were not very careful and the fonts look quite clumsy. This package is usually referred to as a `"really bad cyrillic package for TeX"`.

Nevertheless, we'll discuss it, because it is very easy to use and doesn't require any setup - this collection is supplied with most of TeX distributions.

Of course, you won't be able to use such luxury as automatic hyphenation, but anyway...

1. Prepend your document with the following directives:

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

2. Now to type a cyrillic letter, you enter

```
\cyr
```

and use a corresponding latin letter or a TeX command. Thus, the lower case of the Russian alphabet is expressed by the following codes:

```
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
```

It is extremely inconvenient to convert your Russian texts to such encoding, but you can automate the process. The translit program (section 10.1) supports a TeX output option.

5.2 KOI-8 package for teTeX

There is some new teTeX-rus package. It is reported to support KOI-8 character set and have all basic stuff required for TeX and LaTeX. I personally haven't tried it yes, although I heard about it's successfull usage.

NOTE: This package requires you to reconfigure and rebuild some parts of your **teTeX** package (for example the precompiled LaTeX macros). **Unless you know what you are doing, you shouldn't try it without necessary care. Otherwise, you may be better off by borrowing the precompiled parts fron somebody on the net**

5.3 Using the cmcyralt package for LaTeX

The `cmcyralt` package can be found on any CTAN (Comprehensive TeX Archive Network) site like `ftp.dante.de`. You should obtain two pieces: the fonts collection from `fonts/cmcyralt` and the styles and hyphenation rules from `macros/latex/contrib/others/cmcyralt`.

Note: Make sure you have the `Sauter` package installed, since `cmcyralt` requires some fonts from it. You can get this package from CTAN site as well.

Now you should do the following:

1. Put the new fonts to the TeX fonts tree. On my system (Slackware 2.2) I created a `cmcyralt` directory in the `/usr/lib/texmf/fonts/cm/`. Create the `src`, `tfm`, and `vf` subdirectories in it. Put there `.mf`, `.tfm`, and `vf` files respectively.
2. Put the font driver files (`*.fd`) from the styles archive to the appropriate place (in my case it was `/usr/lib/texmf/tex/latex/fd`).
3. Put the style files (`*.sty`) to the appropriate LaTeX styles directory (in my case `/usr/lib/texmf/tex/latex/sty`).

Now the hyphenation setup. This requires to remake the LaTeX base file.

1. The file `hyphen.cfg` contains the directives for both English and Russian hyphenation. Extract the one for Russian and place it to the LaTeX hyphenation config file `lthyphen.ltx`. In my case, that file was in `/usr/lib/texmf/tex/latex/latex-base`.
2. Put the `rhyphen.tex` to the same directory. It is needed for making the new base file. Later, you can remove it.
3. Do '`make`' in that directory. Don't for get to make a link from `Makefile` to `Makefile.unx`. During the make process check the output. There should be a message:

   ```
   Loading hyphenation patterns for Russian.
   ```

If everything goes OK, you will get the new `latex.fmt` in that directory. Put it to the appropriate place, where the previous one was (like `/usr/lib/texmf/ini/`). **Don't forget to save the previous one!**.

This is it. The installation is complete. Try processing the examples found in the styles archive. If you are to create the PostScript files without any problems, then everything is OK. Now, to use Cyrillic in LaTeX, prepend your document with the following directive:

```
\usepackage{cmcyralt}
```

For more details, see the `README` file in the `cmcyralt` styles archive.

Note: if you do have problems with the examples, provided you have installed the things right, then probably your TeX system hasn't been installed correctly. For example, during my first try, every attempt to create the `.pk` files for the russian fonts failed (`MakeTeXPK` stage). A substantial investigation discovered some implicit conflict between the *localfont* and *ljfour* `METAFONT` configurations. It used to work before, but kept crashing after the `cmcyralt` installation. Contact your local TeX guru - TeX is very (sometimes too much) complicated to reconfigure it without any prior knowledge.

5.4 Using the CyrTUG package

You can obtain the CyrTUG package from the SunSite archive. Get the files `CyrTUGfonts.tar.gz`, `CyrTUGmacro.tar.gz`, and `hyphen.tar.Z`.

The process of installation doesn't differ from too much the previous one.

6 Cyrillic in PostScript

Experts say PostScript is easy. I cannot judge - I've got too many things to learn to spare some time to learn PostScript. So I'll try to use my sad experience with it. **I'll appreciate any feedback from you guys who know more on the subject than I do (approx. 99% of the Earth population).**

Basically, in order to print a Cyrillic text using PostScript, you have to make sure about the following things:

- Cyrillic font is *loaded* or included in the document.
- Cyrillic text is included in the document.
- Cyrillic text uses the appropriate character codes which correspond to the font's requirements.
- An appropriate font is *selected* in order to print Cyrillic text.

There is no solution general enough to be recommended as an ultimate treatment. I'll try to outline various ways to cope with different problems related to the subject.

One way to address Cyrillic setup problems generally enough is to use Ghostscript. **Ghostscript** (or just **gs** in the newspeak) is a free (well quasi-free) PostScript interpreter. It has many advantages; among them:

- Ability to run on many platforms (various Unices, Windows etc)
- Support for a wide number of non-PostScript printers
- Good degree of configurability

What is important in our particular case, is that once **Ghostscript** is set up, we can do all printing through it, thus eliminating extra setup for other PostScript devices (for example *HP LaserJet IV*)

6.1 Adding Cyrillic fonts to Ghostscript

This is important, since you probably don't want to put a responsibility to other programs to insert Cyrillic fonts in the PostScript output. Instead, you add them to **gs** and just make the programs generate Cyrillic output compatible with the fonts.

To add a new font (in `pfa` or `pfb` form) in **gs**, you have to:

1. Put it in the **gs** fonts directory (ie. `/usr/lib/ghostscript/fonts`
2. Add the appropriate names and aliases for the font in the `Fontmap` file in the **gs** directory.

Recently a decent set of Cyrillic fonts for **GhostScript** appeared. It is located in ftp.kapella.gpi.ru. This one even has a necessary part to add to the `Fontmap` file. You have to download the contents of the `/pub/cyrillic/psfonts` directory. The `README` file describes the necessary details.

7 Print setup

Printing is always tricky. There are different printers from different vendors with different facilities. Even for a native printing there is no uniform solution (this applies not only to UNIX, but to other operating systems as well.

Printers have different control languages and often they have very different views on foreign language support. The good news is that on control language seems to be recognized as a de-facto standard for print job description - it is a PostScript language developed by Adobe Corporation.

Another problem is a variety of requirements to the print services. For example, sometimes you want just to print a piece if C program, containing comments in Russian, so you don't need any pretty-printing - just a raw ASCII output in a single font. Another time, when you design a postcard for your girlfriend, you'll probably need to typeset some document with different fonts etc. This will definitely require more effort to setup Cyrillic support.

To accomplish the former task you just have to make your printer understand *one* Cyrillic font and (maybe) install some filter program to generate data in appropriate format. To accomplish the latter one, you have to teach your printer different fonts and have a special software.

There is also something in the middle, when you get a program which knows how to generate both the fonts and the appropriate printer input, so you can say do some aource code pretty-printing without sophisticated word processing systems.

All these options will be more or less covered below.

7.1 Pre-loading Cyrillic fonts into a non-PostScript printer

If you have a good old dot matrix printer and all you need is to print a raw KOI-8 text, try the following:

1. Find a proper KOI-8 font for your printer. Check out the MS-DOSish stuff on the Internet (for example the *SimTel archive* `<ftp://ftp.simtel.net>`).
2. Learn from the manual, how to load such font into your printer and, probably, write a simple program doing that.
3. Run this program from the appropriate `rc` file at a boot time.

Thus, having Cyrillic characters in the upper part of the printer's character set will allow you to print you texts in Russian without any hussle.

Alternatively to the *KOI-8* fonts you may try to use the *Alt* font. There are two reasons for that:

- It may be probably much easier to find an *Alt* font, since those were very widespread in the MS-DOS culture.
- Having a proper *Alt* font will allow you to print pseudo-graphic characters as well.

However in this case, you'll have to convert your texts from *KOI-8* to *Alt* before sending them to a printer. This is quite easy, since there are a lot of programs doing that (see 10.1 for example), so you just have to call such program properly in the `if` field in `/etc/printcap` file. For example, with the **translit** program you may specify:

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

See **printcap(5)** for details.

7.2 Printing with different fonts

One great way to cope with different printers and fonts is to use 5. TeX drivers handle all details, so once you make TeX understand Cyrillic fonts, you are done.

Another possibility is to use *PostScript*. I decided to devote an entire 6 to the subject, since it is not simple.

Finally, there are other word processors, which have printer drivers. I never tried anything apart from TeX, so I cannot suggest anything.

7.3 Converting text to TeX

If all you need is just to print an ASCII text without any additional word processing, you may try to use some programs, which would convert your Cyrillic text to a ready-to-process TeX file. One of the best programs for such purposes is 10.1. In this case, you don't even have to bother about installing the Cyrillic fonts for TeX, since **translit** uses a *Washington Cyrillic* package, which is included in most TeX distributions (or am I wrong?)

7.4 Text to PostScript converters

Sometimes you have just a plain ASCII KOI-8 text and you want to print it just to get it on the paper. One of the easiest ways to achieve that is to use special programs converting text to PostScript.

There are a number of programs doing such conversion. I personally prefer a2ps. Originally developed as a simple text-to-PostScript converter it became a big and highly configurable program with many options and allows you to manage various page layouts, syntax highlighting etc. Another tool (now available as a part of the *GNU* project) is enscript.

The main problem with such programs is that they know nothing about Cyrillic fonts. Right now I am investigating a possibility of including Cyrillic fonts in them in order to understand Cyrillic. Stay in touch.

Nevertheless all the blah-blah above would be pointless without any real advice. So, there we go.

As usual, GNU is the winner. The GNU **enscript** program is designed for converting text to PostScript and has perfect non-ASCII codeset support. It doesn't have Cyrillic PostScript fonts, but it is very easy to get them, as will be explained below (thanks to Michael Van Canneyt):

1. Install the newest **enscript**. As of now, the most recent release is 1.5. You may either get the one from the GNU FTP archive, or take an RPM package from the Redhat site.
2. Now, if you are a lucky RedHat Linux user, download and install *Cyrillic Textbook font* `<ftp://ftp.redhat.com/pub/contrib/i386/enscript-fonts-koi8-1.0-1.i386.rpm>`.
3. If you don't use RPM, download a file `textbook.tar.gz` from the Cyrillic Software collection on *sunsite.unc.edu* `<ftp://sunsite.unc.edu/pub/academic/russian-studies/Software/>`. Extract it to a directory, where **enscript** fonts are located (usually `/usr/share/enscript`). Now change to that directory and run the following command:

   ```
   mkafmmap *.afm
   ```

4. The setup is finished. Try to print some text in KOI8-R Cyrillic with the following command:

   ```
   enscript --font=Textbook8 --encoding=koi8 some.file
   ```

If you don't want a really quick and dirty solution and you don't care about the output quality and all you need is just Cyrillic on the paper, try the rtxt2ps package. It is a very simple no-frills text-to-PostScript conversion program. The output quality is not very good (or, to be honest, just *bad*) but it does it's job.

Another resort is a hacked version of a2ps. This one is quite old, so don't expect all new version's bells and whistles in it. But *it prints Cyrillic text* and the quality is sufficiently better than that of **rtxt2ps**. However, I experienced various kinds of problems with it - like I couldn't print more than two pages (???).

8 Miscellaneous utilities setup

Generally, to set the certain utility up to handle the Cyrillic requires just to allow the 8 bit input. In some cases it is required to tell the application to show the extended ASCII characters in their "native" form.

8.1 bash

Three variables should be set on order to make `bash` understand the 8-bit characters. The best place is `~/.inputrc` file. The following should be set:

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

8.2 csh/tcsh

The following should be set in `.cshrc`:

```
setenv LC_CTYPE iso_8859_5
stty pass8
```

If you don't have the POSIX `stty` (impossible for Linux), then replace the last call to the following:

```
stty -istrip cs8
```

8.3 emacs

The minimal cyrillic support in `emacs` is done by adding the following calls to one's `.emacs` (provided that the Cyrillic character set support is installed for console or X respectively):

```
(standard-display-european t)

(set-input-mode (car (current-input-mode))
   (nth 1 (current-input-mode))
   0)
```

This allows the user to view and input documents in Russian.

However, such mode is not of a big convenience because `emacs` doesn't recognize the usual keyboard commands while set in Cyrillic input mode. There are a number of packages which use the different approach. They don't rely on the input mode stuff established by the environment (either X or console. Instead, they allow the user to switch the input mode by the special `emacs` command and `emacs` itself is responsible for re-mapping the character set. The author took a chance to look at three of them. The `russian.el` package by Valery Alexeev (`ava@math.jhu.edu`) allows the user to switch between cyrillic and regular input mode and to translate the contents of a buffer from one Cyrillic coding standard to another (which is especially useful while reading the texts imported from MS-DOG).

The only inconvenience is that emacs is still treating the russian characters as special ones, so it doesn't recognize russian words' bounds and case changes. To fix it, you have to modify the syntax and case tables of emacs:

```
;; there is a garbage in the variables below, since SGML doesn't like
;; cyrillic characters. You have to put the uppercase and lowercase
;; parts of the Russian alphabet respectively (see the actual files)

(setq *russian-abc-ucase* "*** SGML SUCKS ***")
(setq *russian-abc-lcase* "*** SGML SUCKS ***")


(let ((i 0)
      (len (length *russian-abc-ucase*)))

     (while (< i len)
       (modify-syntax-entry (elt *russian-abc-ucase* i) "w  ")
       (modify-syntax-entry (elt *russian-abc-lcase* i) "w  ")
       (set-case-syntax-pair (elt *russian-abc-ucase* i)
                             (elt *russian-abc-lcase* i)
                             (standard-case-table))
       (setq i (+ i 1))))
```

For this purpose I created a `rusup.el` file which does this, as well as a couple handy functions. You have to load it in your `~/.emacs`.

Another alternative is the package `remap` which tries to make such support more generic. This package is written by Per Abrahamsen (`abraham@iesd.auc.dk`) and is accessible at `ftp.iesd.auc.dk`.

As for the author's opinion, I would suggest to start using the `russian.el` package because it is very easy to setup and use.

8.4 ispell

There is an rspell add-on created by Neal Dalton (`nrd@cray.com`) for the **GNU ispell** package, but I experienced some problems making it work right away. Try it - maybe you will be luckier.

9 joe

Try the `-asis` option.

9.1 ksh

As for the public domain `ksh` implementation - `pdksh 5.1.3`, you can input 8 bit characters only in `vi` input mode. Use:

```
set -o vi
```

9.2 less

So far, `less` doesn't support the KOI-8 character set, but the following environment variable will do the job:

```
LESSCHARSET=latin1
```

9.3 lynx

As of version 2.6, you may select the appropriate value for the `display Character set` opetion.

9.4 mc (The Midnight Commander)

To display Cyrillic text correctly, select the *full 8 bits* item in the **Options/Display** menu.

If your problem is the ugly windows' borders, consult the 3.1 section.

As an off-topic, if you want to make **mc** use color in an `Xterm` window, set the variable `COLORTERM`:

```
COLORTERM= ; export COLORTERM
```

9.5 Netscape navigator

Make sure you are using `Netscape` version higher than 3. If your `Netscape` is older, download a new one from www.netscape.com.

9.5.1 Basic setup

To be able to see Cyrillic text in most parts of the HTML document, do the following:

- In menu **Options/Document Encoding** select **Cyrillic(KOI-8)**.
- In menu **Options/General Preferences/Fonts** select **Cyrillic (KOI-8)** encoding, **Times(Cronyx)** as a proportional font and **Courier(Cronyx)** as a fixed one.
- save options.

NOTE: This setup will work with most parts of the document. However, you won't be able to display Cyrillic text in the window header, menus and some controls. To fix these problems, do an

9.5.2 Advanced setup

Andrew A. Chernov is the one, who knows more than others about KOI-8 in general and netscape in particular. Visit his excellent KOI-8 page and download a patch for Netscape resource file, making Netscape speak Russian as much as it is able to.

9.6 pine

Set the following directive in `~/.pinerc` for personal configuration, or in `/usr/lib/pine.conf` for a global one:

```
character-set=ISO-8859-5
```

9.7 rlogin

Make sure that the shell on the destination site is properly set up. Then, if your `rlogin` doesn't work by default, use '`rlogin -8`'.

9.8 sendmail (aka "The Doom of a Sysadmin")

As of version 8, `sendmail` handles 8-bit data correctly by default. If it doesn't do it for you, check the `EightBitMode` option and option `7` given to mailers in your `/etc/sendmail.cf`. See *"Sendmail. Operation and Installation Guide"* for details.

9.9 zsh

Use the same way as with `csh` (see section 8.2). The startup files in this case are `.zshrc` or `/etc/zshrc`.

10 Useful Tools

10.1 Conversion Utilities

There are a number of programs able to convert from *KOI-8* to *Alt* and back. Look at SovInformBureau or ftp.funet.fi for a list of handy little utilities. You can even use the special mode for `emacs` (see section 8.3).

However, I would especially recommend a translit package. It supports many popular codesets and is even able to produce a *TeX files (see section 5) from text in Russian. Also, RedHat users will enjoy an RPM package for translit.

10.2 Programmer's tools

So far, I explained the ways to make the programs accept and display the Cyrillic codeset. However the full *localization* of the system comprises much more. All discussed above is not enough. The system should be friendly for a user who doesn't necessarily speak English. In my own opinion, it is not a big deal to become familiar with English at the level of the programs' messages. However, it is not quite fair to require it.

Thus, the next level of localization requires the programs to be customizable to the requirements of different languages and data representation habits.

Before, that was done by developing some abstraction of the messages to output from the program's code. Now, such mechanism is (more or less) standardized. And, of course, there are free implementations of it!

The good news is that GNU finally adopted the way of making the internationalized applications. Ulrich Drepper (`drepper@ipd.info.uni-karlsruhe.de`) developed a package `gettext`. This package is available at all GNU sites like prep.ai.mit.edu. It allows you to develop programs in the way that you can easily make them support more languages. I don't intend to describe the programming techniques, especially because the `gettext` package is delivered with excellent manual.

So, if you are developing programs which output messages (have you ever developed any program which didn't?), then don't be lazy to put a little (yes, really little) effort to make your program locale-aware.

Request for collaboration: If you want to learn the `gettext` package and to contribute to the GNU project simultaneously; or even if you just want to contribute, then you can do it! GNU goes international, so all the utilities are being made locale-aware. The problem is to translate the messages from English to Russian (and other languages if you'd like). Basically, what one has to do is to get the special `.po` file consisting of the English messages for a certain utility and to append each message with it's equivalent in Russian. Ultimately, this will make the system speak Russian if the user wants it! For more details and further directions contact Ulrich Drepper (drepper@ipd.info.uni-karlsruhe.de).

11 Summary of the various useful resources

a2ps homepage `<http://www-inf.enst.fr/ demaille/a2ps.html>`

A. Chernov's KOI-8 page `<http://www.nagual.ru/ ache/koi8.html>`

General Linux Information <http://sunsite.unc.edu/mdw/linux.html>

My collection of stuff related to the Cyrillic setup
<ftp://netvision.net.il/pub/home/b/belikoff/cyrillic/>

Collection of Cyrillic stuff on ftp.kiae.su <ftp://ftp.kiae.su/cyrillic/>

Collection of Cyrillic stuff on ftp.relcom.ru <ftp://ftp.relcom.ru/cyrillic/>

Collection of cyrilization software <ftp://ftp.funet.fi/pub/culture/russian/comp/>

Cronyx <http://www.cronyx.ru> - the creators of Cyrillic fonts for the X Window System.

Cyrillic fonts for Ghostscript <ftp://ftp.kapella.gpi.ru/pub/cyrillic/psfonts>*Cyrillic fonts for X*
<ftp://ftp.kiae.su/cyrillic/x11/fonts/xrus-2.1.1-src.tgz>

Ghostscript <http://www.cs.wisc.edu/ ghost/index.html>

GNU nenscript <ftp://prep.ai.mit.edu/pub/gnu>

Information on Cyrillic Software <http://sunsite.oit.unc.edu/sergei/Software/Software.html>

relcom.fido.ru.unix <news:relcom.fido.ru.unix> newsgoup.

RFC 1489 <file://ds.internic.net/rfc/rfc1489.txt>

rspell for GNU ispell
<ftp://sunsite.unc.edu/pub/academic/russian-studies/Software/rspell.tar.gz>

SovInformBureau <http://www.siber.com/sib/russify/>

teTeX russification package <ftp://xray.sai.msu.su/pub/outgoing/teTeX-rus/>

The kbd package for Linux <ftp://sunsite.unc.edu/pub/Linux/system/Keyboards/>

The remap package for Emacs <ftp://ftp.iesd.auc.dk/>

The rtxt2ps and hacked a2ps packages <http://www.siber.com/sib/russify/converters/>

The translit package <ftp://ftp.osc.edu/pub/russian/translit/translit.tar.Z>

The xruskb package <ftp://ftp.relcom.ru/x11/cyrillic/>

Useful Cyrillic packages <ftp://sunsite.unc.edu/pub/academic/russian-studies/Software>

X fonts collections <ftp://ftp.switch.ch/mirror/linux/X11/fonts/>

XFree86 FTP site <http://www.xfree86.org>