The Linux Cyrillic HOWTO

Alexander L. Belikoff

v2.1, 17 September 1995

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

Contents

1 Introduction

[27 June 1996
As far as I am concerned, this HOWTO is not being maintained by the author any more. If you are interested in maintaining the Cyrillic HOWTO, or know how to contact the author, please get in touch with me at gregh@sunsite.unc.edu]

This document covers the things you need to successfully typeset, view, and print documents in Russian.

The description is targeted primarily to the end users. While writing it, I tried to keep the things as simple as possible. On my opinion, it is unfair to overburden the user with the X Window System or TeX implementation details, because she merely wants to type or view the nice texts in Russian. However, I don't describe *what* the X Window System is and *how* to typeset the documents with TeX and LaTeX. Consequently, having a guru at hand for the case of potential problems would be an advantage.

There is some conflict between MS-DOG and Un*x Cyrillic codesets. In MS-DOG, the most popular Cyrillic codeset is Alt (so-called alternative codeset). In Un*x, however, the traditional codeset with Russian characters is KOI-8. It is specified in the RFC 1489 (*"Registration of a Cyrillic Character Set"*).

The difference in two codesets is usage is pretty minimal (except the TeX packages). Therefore, we will describe only KOI-8 codeset setup.

I appreciate any comments corrections and suggestions concerning the document.

2 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 have forgotten to mention somebody.

So, here they go: Bas V._de Bakker, David Daves, Serge Vakulenko, Sergei O. Naoumov, Winfried Truemper.

3 Further plans

The next versions of this document will be accessible on `sunsite.unc.edu` and `tsx-11.mit.edu` in HOWTO directory of the Linux Document Project.

I am looking forward to include more information on TeX and LaTeX as well as much more elaborated information on printing.

4 Setting up the environment

4.1 Console

All stuff needed for russification of the Linux console is contained in the kbd package. The package is accessible at `sunsite.unc.edu` or `tsx-11.mit.edu`. Usually, that package is already installed (it is a standard part of at least Slackware distribution).

To setup the Cyrillic stuff, one should do three things:

1. Set the appropriate screen font. This is performed by the `setfont` program. The fonts files are placed in `/usr/lib/kbd/consolefonts`.

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

2. If you use the font in Alt coding (as I do) then you have to set up the screen mapping program to perform automatic conversion from Alt to KOI-8. For that purpose use the `mapscrn` program and the `/usr/lib/kbd/consoletrans/koi2alt` file.

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

4. Output an *ESC*(K escape sequence on the screen (*ESC* stands for the Escape character with code 033). Only God knows the purpose of that combination. I stole it from the *Danish-HOWTO* `<file://sunsite.unc.edu/pub/Linux/docs/HOWTO/Danish-HOWTO>` (thanks, Thomas Petersen) and it works for me!

The following is an example of a script which sets up the Cyrillic mode for console:

```
#!/bin/bash
#
# load cyrillic defs for console
#
# *** NEVER TRY IT UNDER X!!! ***

loadkeys /usr/lib/kbd/keytables/ru.map
setfont /usr/lib/kbd/consolefonts/Cyr_a8x16
mapscrn /usr/lib/kbd/consoletrans/koi2alt
echo -ne 'ESC(K'        # use the REAL ESCAPE character here !
echo "Use the right Ctrl key to switch the mode..."
```

4.2 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.

4.2.1 The X fonts.

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

There is a number of such fonts on the net. The author's favorite one is the collection `VakuFonts` created by Serge Vakulenko (`vak@cronyx.ru`). It can be found in the collection of *cyrillic stuff for the X Window System* `<ftp://ftp.funet.fi/pub/culture/russian/comp/xwin>` where you can find many useful packages for X.

Note: Apparently, that package is included to the XFree86 version 3.1.2 as well as to the most recent public patch for the X11 Release 6. Unfortunately, the author hasn't had a chance to check it yet.

Usually the X fonts are distributed in the `BDF` format which is actually the textual font description. You should compile the fonts to the `PCF` format using the `bdftopcf` command:

```
bdftopcf -o name.pcf name.bdf
```

It is also possible to compress the compiled font using the `compress` program (I am not sure about the `gzip` support).

Now you should do three things to set everything up:

1. Put the compiled (and possibly compressed) fonts to the specified directory.

2. Recreate the list of fonts for the directory. Simply `cd` to it and run:

   ```
   mkfontdir .
   ```

   You should run it once. This will upgrade the the fonts catalog file `fonts.dir`.

3. If the fonts package provides the file of fonts' aliases (usually `fonts.alias`) then append it's contents to the fonts' aliases file in the directory containing the fonts.

4. If that directory is not already known to the X server, then you should make it known. To achieve that, add the following commands to the `xinitrc` file (either local or global one):

   ```
   xset +fp directory_with_fonts xset fp rehash
   ```

After you have made the settings above, you can check the availability of the new fonts by running the following command:

```
xfd -fn fontname
```

This should show the table of characters of the specified font.

4.2.2 The input translation

The switching between the different input translations is set up by the `xmodmap` program. This 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, usually `~/.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 X11 Release 7 which will 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

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 *syms* 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 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 haven't used 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 have 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've included 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
```

The mode switching rules This is basically 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.

There are two ways to achieve that in Linux. One is XFree86-specific, while the other is more general (well, not too much, as I'll show below).

The XFree86-specific way is the following. There are two virtual actions which can be assigned to the keys in the `XF86Config` file: `ModeShift` which changes to the mode alternative to the regular one without locking, and `ModeLock` which does the same but with locking. In the first case the keys will emit the alternative keysyms only when the key generating the `ModeShift` is held down, whereas in the latter case the user needs to press the key generating the `ModeLock` keysym only once and the keyboard will be generating the alternative keysyms until that key is pressed for a second time. You should assign the `ModeShift` and `ModeLock` keysyms to the keys you want to work the mode switches.

Thus, if one wants to assign the `ModeShift` action to the right Alt key, she should place the following directive in her XF86Config:

```
RightAlt        ModeShift
```

Similarly, if the action required was ModeLock, the directive would be:

```
RightAlt        ModeLock
```

See the **XF86Config(4/5)** for more details.

The other way is, again, to use the `xmodmap` utility. This is much more tricky. Basically 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.

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 perfect 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 `.xinitrc` file to perform it. **NOTE:** your `.xinitrc` can already contain the code to run the `xmodmap` over your local table if the one exists.

The table distributed with the Serge's Vakulenko package didn't work for the author. The following patch fixed the problem:

```
diff -u --new-file jcuken.xmm jcuken.xmm.mod
--- jcuken.xmm  Mon May 20 09:11:36 1991
+++ jcuken.xmm.mod      Sun Aug 13 15:44:06 1995
@@ -2,6 +2,8 @@
 ! Cyrillic keyboard mapping table.
 ! Produced by Serge Vakulenko, <vak@kiae.su>, Moscow.
 !
+! Modified by Alexander L. Belikoff (abel@wisdom.weizmann.ac.il), 1995
+!
 ! Russian JCUKENG keyboard layout implemented.
 ! Cyrillic characters are entered in koi8 encoding.
 !
@@ -10,7 +12,9 @@

 ! Use CapsLock as rus/lat switch key.
 remove lock = Caps_Lock
-add mod2 = Caps_Lock
+keysym Caps_Lock = Mode_switch
+add mod2 = Mode_switch
+add lock = Mode_switch

 !       Key             Base            Shift           Caps    ShiftCaps
 !------------------------------------------------------------------------
```

This allowed me to use the Caps Lock key to switch between normal and Cyrillic input modes.

5 Cyrillic support in LaTeX

In this section I'll describe the procedure of making LaTeX typeset Russian. LaTeX is a macro package for TeX offering the user many useful styles, templates, and commands. If you are making your mind on what package to use, I personally suggest LaTeX. Moreover, 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. So far I describe the LaTeX 2e setup.

I have an experience with two 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 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: 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 8.1.

5.1 Using the cmcyralt package

The `cmcyralt` package can be found on any CTAN (Comprehensive TeX Archive Network) site like `ftp.dante.de`. You should get two collections: 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.

5.2 Using the CyrTUG package

You can obtain the CyrTUG package from the *SunSite archive* `<ftp://sunsite.unc.edu/pub/academic/russian-studies/Software>`. Get the files `CyrTUGfonts.tar.gz`, `CyrTUGmacro.tar.gz`, and `hyphen.tar.Z`.

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

6 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.

6.1 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
```

6.2 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
```

6.3 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 `rustable.el` (sorry, I don't know the author of it) adds the syntax rules of Cyrillic codeset to `emacs` (words' bounds, case change rules etc.) These packages can be found at most Emacs-Lisp archives. Another one 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.

6.4 ispell

Check the `sunsite.unc.edu:/pub/academic/russian-studies/Software` for the russian dictionary created by Neal Dalton (`nrd@cray.com`) for the `ispell` package.

6.5 less

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

```
LESSCHARSET=latin1
```

6.6 Netscape

Set the following resource:

```
*documentFonts*registry:        koi8
```

6.7 rlogin

Use '`rlogin -8`'

7 Printing

To print the text files containing the Russian characters using on PostScript printers, you need two things: the fonts and the 8bit-aware software able to print the texts using those fonts. The best package to print the text files in PostScript is `a2ps` by Evan Kirshenbaum (`evan@csli`) and Miguel Santana (`miguel@imag.imag.fr`). The last version is 8bit-aware. You can get it from `imag.imag.fr:/archive/postscript/`.

Check the `sunsite.unc.edu:/pub/academic/russian-studies/Software/` in order to obtain the Cyrillic PostScript fonts. Also, there is a lot of fonts in the *SimTel* `<ftp://oak.oakland.edu/pub/SimTel/>` collection.

8 Useful Tools

8.1 User's tools

There are number of programs able to convert from *KOI-8* to *Alt* and back. You can even use the special mode for `emacs` (see section 6.3).

One nice standalone package is `translit`. It is available at the *SunSite archive* `<ftp://sunsite.unc.edu/pub/academic/russian-studies/Software>`. This package is capable of converting between different formats including the *KOI-8* and the *Alt* ones.

8.2 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* `<ftp://prep.ai.mit.edu/pub/gnu/>`. 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* `<mailto:drepper@ipd.info.uni-karlsruhe.de>`).

9 Summary of the various useful resources

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

Many fonts collections for X `<ftp://ftp.funet.fi/pub/culture/russian/comp/xwin>`

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

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

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

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

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