churchslavonic package — Church Slavonic Typography in LaTeX

Aleksandr Andreev and Mike Kroutikov

version v0.2.4

June 24, 2023

Abstract

Package churchslavonic provides fonts, hyphenation patterns and supporting macros to typeset Church Slavonic texts.

Contents

Introduction

Church Slavonic (also called Church Slavic, Old Church Slavonic or Old Slavonic; ISO 639-2 code `cu`) is a literary language used by the Slavic peoples; presently it is used as a liturgical language by the Russian Orthodox Church, other local Orthodox Churches, as well as various Byzantine-Rite Catholic and Old Ritualist communities. The package `churchslavonic` provides fonts, hyphenation patterns and supporting macros to typeset Church Slavonic texts in TEX.

The package is designed to support Unicode text encoded in UTF-8. Texts encoded in legacy codepages (such as HIP and UCS) may be converted to Unicode using a separate bundle of utilities. See the Slavonic Computing Initiative website for more information. To use the tools in this package, you will need a Unicode-aware TEX engine such as XƎTEX or LuaTEX.

1 How to use the package

To use the churchslavonic package one needs to include the following declarations into the document preamble:

```
\usepackage{polyglossia}
\setmainlanguage{churchslavonic}
\usepackage{churchslavonic}
```

This makes the Church Slavonic hyphenation patterns immediately available. After that, use the standard polyglossia commands to control current language. Church Slavonic fonts are provided by the fonts-churchslavonic package, which should have been installed automatically when you installed this package. See the fonts-churchslavonic documentation for information about fonts.

1.1 Options `color`, gray (= grey), and `bw`

These options control what color is actually being used for text coloring commands like `\cuKinovar`.

- `color` this is the default option; it indicates the original color (a shade of red).
- `gray` (`grey`) replaces the red color with gray - useful if you are printing on media that does not support color, but is capable of grayscale.
- `bw` replaces the red color with black (effectively turning off commands like `\cuKinovar` and `\cuKinovarColor`). Use this option to generate a document that will be printed in black-and-white.

Example:

```
\usepackage[gray]{churchslavonic}
```

1.2 Underscore

The underscore symbol (_, U+005F Low Line) is a valid text symbol in Church Slavonic (it has roughly the same role as the hyphen in English). The package churchslavonic redefines the underscore in a way that it can be directly entered in text mode, for example:

> Послѣ́дованїе моле́бнагѡ пѣ́нїѧ ст҃ы̑мъ мч҃камъ к҃_гѡ вѣ́ка,
> въ Са́нктъ_Петербꙋ́ржстѣй дꙋхо́внѣй а҆каде́мїи
> нача̑льствовавшимъ, ѹ҆чи̑вшимъ и҆ ѹ҆чи̑вшимсѧ

Attention: if you have an older version of the fontspec package installed on your system, the redefined underscore symbol cannot be used in font names and font options in fontspec commands like `\setXXXfont` and `\newfontfamily`.

Typically you need to set underscore as the hyphenation character for Church Slavonic fonts: `HyphenChar=_`. With older versions of fontspec this will cause errors. This problem exists in TEX `Live 2013` and in fontspec v2.3c.

We recommend upgrading your TEX distribution to at least TEX `Live 2015`. Alternatively you can selectively upgrade the fontspec package to version `v2.4c` or better.

If upgrading is not an option, you can work around this problem by either specifying the hexadecimal code for the hyphenation character: `HyphenChar="005F`, or by declaring all fonts before loading churchslavonic.

2 Numbers

The Church Slavonic numbering system (Cyrillic numerals) is based on Greek Ionian numerals and uses letters as digits. For more information on the implementation, consult the appropriate section of [1].

2.1 `\cuNum`

Use this command to typeset a Cyrillic numeral. The command takes a single argument that should expand to a number.

`\cuNum{1}`	а҃
`\cuNum{12}`	в҃і
`\cuNum{123}`	рк҃г
`\cuNum{1234}`	҂асл҃д
`\cuNum{10345}`	҂ітм҃є
`\cuNum{12345}`	҂в҃і тм҃є
`\cuNum{123456}`	҂рк҃г ун҃ѕ
`\cuNum{800456}`	҂ѿун҃ѕ
`\cuNum{1234567}`	҂҂а҃ ҂сл҃д фѯ҃з
`\cuNum{1500567}`	҂҂а҃ ҂ффѯ҃з
`\cuNum{12345678}`	҂҂в҃і ҂тм҃є хѻ҃и
`\cuNum{123456789}`	҂҂рк҃г ҂ун҃ѕ ѱп҃ѳ

3 Dates

`\cuDate{2016-4-21}`	к҃а а҆прі́ллїа, лѣ́та ҂вѕ҃і
`\cuDateJulian{2016-4-21}`	и҃ а҆прі́ллїа, лѣ́та ҂вѕ҃і
`\cuDate{\cuToday}`	к҃д і҆ꙋ́нїа, лѣ́та ҂вк҃г

3.1 `\cuDate`

This command formats the date (according to the current format). The argument is a triplet of numbers `YYYY-MM-DD` specifying the date. The output will be something like this: к҃в а҆прі́ллїа, лѣ́та ҂вѕ҃і.

Note that `YYYY-MM-DD` values are not being normalized or interpreted in any way. Thus, it is totally fine to call `\cuDate{2016-4-32}` even though April 32 is not a valid date. It will be formatted and printed as April 32. This makes it possible to use this macro in a phrase like “the date `\cuDate{2016-4-32}` is not a valid date in any calendar”.

However, if your date format uses `\cuDOW` (day of the week) or `\cuYEARAM` (year Anno Mundi), the later quantities are computed by interpreting the date as a Gregorian calendar date. In this case, if the input date is not a valid date, it will be normalized via extrapolation. For example, April 32 will be interpreted as May 2 for the purpose of determining values of the day of the week and year Anno Mundi.

If your format uses `\cuDOW` or `\cuYEARAM`, and you specify a date according to the Julian, not Gregorian, calendar, you must use `\cuDateJulian` to correctly format days of the week and year Anno Mundi.

The best practice is to always use `\cuDate` with Gregorian calendar dates and use `\cuDateJulian` with Julian calendar dates regardless of the current date format. This way you can switch the date formatting style without worrying about getting the wrong output.

3.2 `\cuDateJulian`

Formats the date, just like `\cuDate` does, but the argument is interpreted as a date on the Julian calendar ("old style") instead of the Gregorian calendar. This makes a difference only if your format is using symbolic names `\cuDOW` and/or `\cuYEARAM`.

3.3 `\cuDefineDateFormat`

This command allows you to define your own date format. It does not change how `\cuDate` formats its output (for that, use `\cuUseDateFormat`). Example:

```
\cuDefineDateFormat{long}{%
  \cuDayName{\cuDOW},
  \cuNum{\cuDAY}\textunderscore гѡ~%
  \cuMonthName{\cuMONTH},~%
  лѣ́та ѿ сотворе́нїѧ мі́ра~%
  \cuNum{\cuYEARAM}%
}
```

defines a format with name `long`. If we use this format to print the same date as before, we will get: пѧто́къ, к҃в_гѡ а҆прі́ллїа, лѣ́та ѿ сотворе́нїѧ мі́ра ҂зфк҃є.

The following symbolic names can be used when formatting the date:

- `\cuYEAR` — the year part of a date (a number, like `2016`)
- `\cuYEARAM`[1] — the year Anno Mundi, that is, since the creation of the world accoding to the Byzantine reckoning (aka "the Byzantine era"; a number, like `7525`)

- `\cuMONTH` — the month part of a date (a number from 1 to 12, with January set to 1)
- `\cuDAY` — the day of the month
- `\cuDOW`[1] — the day of the week (number from 0 to 6, where 0 means "Sunday")
- `\cuINDICTION` — the indiction[2] (a number from 1 to 15)

3.4 `\cuUseDateFormat`

This command sets the date format to be used by the subsequent `\cuDate` and `\cuDateJulian`.

3.5 `\cuMonthName`

This command expands a numeric argument (month number) into textual representation. It is typically used when defining a date format. For example, a date format named `default` is defined as:

```
\cuDefineDateFormat{default}{%
  \cuNum{\cuDAY}~\cuMonthName{\cuMONTH},%
  ~лѣ́та~\cuNum{\cuYEAR}%
}%
```

3.6 `\cuDayName`

Expands a numeric argument into a textual representation of the day of the week in the nominative case.

3.7 `\cuDayNameAccusative`

Expands a numeric argument into a textual representation of the day of the week in the accusative case.

3.8 `\cuToday`

This macro expands to a triplet `YYYY-MM-DD`. The date is generated according to the Gregorian calendar.

---

[1] If your format uses this value, make sure that you format the date with the correct macro: you must use `\cuDate` for dates on the Gregorian calendar and `\cuDateJulian` for dates on the Julian calendar.

[2] See https://en.wikipedia.org/wiki/Indiction.

3.9 `\cuTodayJulian`

This macro expands to a triplet `YYYY-MM-DD`. The date is generated according to the Julian calendar.

It is a shortcut for `\cuAsJulian{\cuToday}`.

3.10 `\cuAsJulian`

Converts a date on the Gregorian calendar to a date on the Julian calendar. Input and output use numeric triplet format `YYYY-MM-DD`.

Useful when the same date needs to be formatted both according to the Gregorian and Julian calendars.

3.11 `\cuAsGregorian`

Converts a date according to the Julian calendar to a date according to the Gregorian calendar. Input and output use numeric triplet format `YYYY-MM-DD`.

4 Kinovar

Printed and hand-written Church Slavonic texts often use color to highlight sectional and paragraph structure and to indicate liturgical rubrics, section names, comments, and marginal notes. The first letter of each paragraph is also often colored red.

4.1 `\cuKinovar`

Takes a single argument and prints it using red color. For example, explicitly specifying its argument one gets the expected result:

`\cuKinovar{ли́къ:} гдⷭ҇и поми́лꙋй.`	ли́къ: гдⷭ҇и поми́лꙋй.

If one uses the TEX mechanism of implicit argument detection, then the first character of the text after this command will be printed in red. However, a non-trivial feature of this command is that it will also "collect" all of the diacritical marks that belong to this first character, and thus all accents will also be colored in red! Use this command in the indirect parameter mode to print in red the first letter of each paragraph.[1]

---

[1] You can also experiment with the TEX command `\everypar` to automate this, but the success or failure of this technique critically depends on the LATEX class used and packages loaded. We found it very fragile and thus the churchslavonic package does not offer any automation for this task. It

`\cuKinovar` Пои́мъ гдⷭ҇ви пѣ́снь но́вꙋю	Пои́мъ гдⷭ҇ви пѣ́снь но́вꙋю
`\cuKinovar` Ꙗ҆́кѡ тꙋ́ча на тро́скотъ	Ꙗ҆́кѡ тꙋ́ча на тро́скотъ

4.2 `\cuKinovarColor`

Switches the current color to red. One would typically use this command inside a group that limits the scope of red text, unless you want all subsequent text to be colored red.

Information: The shade of red used by the `\cuKinovar` is declared in the package as a new color called `kinovar`, and is set to (205, 8, 3) in the RGB color space, which has a hexadecimal representation of `#CC0502`.

5 Miscellaneous

5.1 `\cuMarginMark`, `\cuMarginMarkSkip`, & `\cuMarginMarkText`

The command `\cuMarginMark` is used to place short text in the margin at the same level as the line where the macro is placed. Example:

```
\cuMarginMark{в҃}\cuKinovar Бл҃гословѝ дꙋшѐ моѧ̀ гдⷭ҇а и҆ не забыва́й
всѣ́хъ воздаѧ́нїй є҆гѡ̀. \cuMarginMark{д҃}\cuKinovar Ѡ҆чища́ющаго
всѧ̑ беззакѡ́нїѧ твоѧ̑, и҆сцѣлѧ́ющаго всѧ̑ недꙋ́ги твоѧ̑:
\cuMarginMark{в҃}\cuKinovar И҆збавлѧ́ющаго ѿ и҆стлѣ́нїѧ живо́тъ тво́й,
вѣнча́ющаго тѧ̀ млⷭ҇тїю и҆ щедро́тами: \cuMarginMark{д҃}\cuKinovar
И҆сполнѧ́ющаго во бл҃ги́хъ жела́нїе твоѐ, ѡ҆бнови́тсѧ ꙗ҆́кѡ ѻ҆́рлѧ
ю҆́ность твоѧ̀.
```

Will result in:

в҃ Бл҃гословѝ дꙋшѐ моѧ̀ гдⷭ҇а и҆ не забыва́й всѣ́хъ воздаѧ́нїй
д҃ є҆гѡ̀. Ѡ҆чища́ющаго всѧ̑ беззакѡ́нїѧ твоѧ̑, и҆сцѣлѧ́ющаго
в҃ всѧ̑ недꙋ́ги твоѧ̑: И҆збавлѧ́ющаго ѿ и҆стлѣ́нїѧ живо́тъ
д҃ тво́й, вѣнча́ющаго тѧ̀ млⷭ҇тїю и҆ щедро́тами: И҆сполнѧ́ющаго
во бл҃ги́хъ жела́нїе твоѐ, ѡ҆бнови́тсѧ ꙗ҆́кѡ ѻ҆́рлѧ ю҆́ность
твоѧ̀.

---

may be easier and more robust to use the Find/Replace functionality embedded in any non-trivial text editor to just automatically place a `\cuKinovar` command before every paragraph of the source text.

The marginal mark is placed on the right margin for odd pages and on the left margin for even pages (e.g. the mark is placed on the “outer” margin, not the spine margin), which is usualy the desired behavior.

The distance between the mark and the text is controlled by the value of `\cuMarginMarkSkip`. The default is:

```
\def\cuMarginMarkSkip{0.6em}
```

To globally customize the font and color of the margin mark use `\cuMarginMarkText`. For example, to make margin marks appear in red color, redefine `\cuMarginMarkText` in the preamble of your document like this:

```
\def\cuMarginMarkText#1{\cuKinovar{#1}}
```

If you need to change the font/size/color just for a single mark, you can do it directly with `\cuMarginMark`:

```
\cuMarginMark{{\tiny *}}
```

5.2 Superscripts

The `\cuSup` macro can be used to place arbitrary superscript text above the baseline text, a feature that can be useful in editing Slavonic manuscripts. The macro takes two parameters: the first parameter is the superscripted text, the second parameter is the baseline text. For example:

```
мол\cuSup{ва}{и◌́т}
```

$\overset{\text{ва}}{\text{молит}}$

The macro accepts an optional parameter `raise`, which controls the vertical space between the base characters and the superscript. The formatting of the base and superscript texts can also be controlled directly:

```
по\cuSup[raise=0.75ex]{\kern2em \cuKinovar{лага◌́етъ}}{\cuKinovar{ложй}} на◌́мъ
```

$\text{по}\overset{\text{лага́етъ}}{\text{ложѝ}}\ \text{на́мъ}$

5.3 Dropcaps

The mechanism that the `\cuKinovar` macro is using to collect all accents (when the argument is specified implicitly) can be useful for many other purposes. One example is to typeset a dropletter at the beginning of a chapter (this is often used in Church Slavonic texts). For this purpose, the standard LaTeX package `lettrine` works just fine. The only nuisance is that one has to be careful to pass to `\lettrine` not just the first letter, but also any diacritical marks that attach to this letter. Naturally, we want to reuse the clever mechanism that `\cuKinovar` uses, and automatically collect the diacritical marks!

Here is an example of how to accomplish this:

```
\def\cu@lettrine{\lettrine[lines=3,findent=0pt,nindent=0pt]}
\def\cuLettrine{\cu@tokenizeletter\cu@lettrine}
\renewcommand{\LettrineFontHook}{\cuKinovarColor}
```

Once this definition of `\cuLettrine` is created (somewhere in the preamble, between the declarations `\makeatletter` and `\makeatother`), you can create drop capitals like this:

`\cuLettrine` И҆́же дх҃а си́ла въ не́мощи соверша́етсѧ...

И҆́же дх҃а си́ла въ не́мощи соверша́етсѧ, ꙗ҆́коже пи́сано є҆́сть, и҆ вѣ́рꙋемъ: въ не́мощи же не тѣлесѐ то́чїю, но ᲂу҆́бѡ и҆ сло́ва, и҆ премꙋ́дрости на ѧ҆зы́цѣ лежа́ща. И҆ сѐ ꙗ҆́вѣ ѿ мно́гихъ ᲂу҆́бѡ и҆ны́хъ, па́че же ѿ и҆́же ѡ҆ вели́комъ бг҃осло́вѣ, и҆ бра́тѣ хрⷭ҇то́вѣ, благода́тїю зри́мѣмъ.

6 Znamenny Notation

6.1 Typesetting Znamenny Notation

The package offers two commands for typesetting liturgical music in Znamenny Notation and other neumatic notation systems. To use these macro commands, you will need to declare the font that will be used to typeset the musical notation symbols and set the `\cuKrukFont` parameter, which is usually done in the preamble of the document. Example:

```
\newfontfamily\musicFont[Scale=1.5]{Mezenets Unicode}
\let\cuKrukFont=\musicFont
```

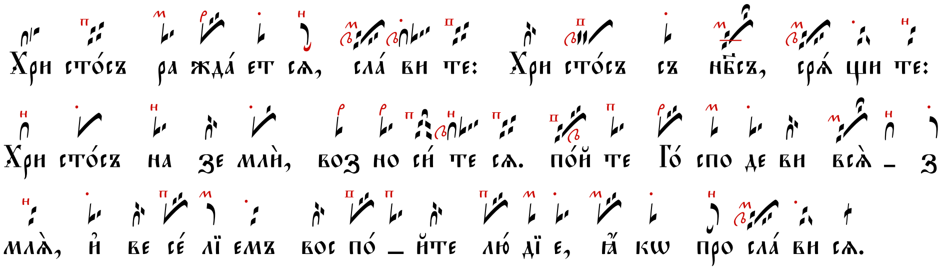
The command `\cuKruk` is used to typeset a syllable of text with a musical neume above it. It takes the neume as its first argument and the text as its second argument. This can be used to typeset Znamenny notation inline with text or to typeset relatively short passages, for example, the code:

```
Here is a Podchashie: \textchurchslavonic{\cuKruk{[illegible]}{Тво}}
```
will produce:

Here is a Podchashie: Тво

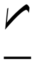
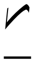
The `\cuKrukPara` command is used to typeset longer, independent passages of music in Znamenny Notation. The command takes as an argument a line of neumes, followed by \\, followed by a line of lyrics. Groups of neumes are separated by spaces and syllables are separated by the hyphen (`-`) character. Here we use the command to typeset a piece in Znamenny Notation:

```
\cuKrukPara{[illegible]
[illegible]
[illegible]
[illegible] \\
Хри-сто́съ ра-жда́-ет-сѧ, сла́-ви-те: Хри-сто́съ съ нб҃съ, срѧ́-щи-те:
Хри-сто́съ на зе-млѝ, воз-но-си́-те-сѧ. по́й-те Го́-спо-де-ви всѧ̀-~ зе-
млѧ̀,
и҆ ве-се́-лї-емъ вос-по--йте лю́-дї-е, ꙗ҆́-кѡ про-сла́-ви-сѧ.}
```


Хри стосъ ра жда́ ет сѧ, сла́ ви те: Хри стосъ съ нб҃съ, срѧ́ щи те:
Хри стосъ на зе млѝ, воз но си́ те сѧ. по́й те Го́ спо де ви всѧ̀ зе
млѧ̀, и҆ ве се́ лї емъ вос по́ йте лю́ дї е, ꙗ҆́ кѡ про сла́ ви сѧ.

Note 1: the number of neume groups needs to equal the number of syllables, otherwise the `\cuKrukPara` command will return a compile-time error, such as: `! Too many kruk groups.`

Note 2: In both the `\cuKruk` and `\cuKrukPara` commands, the ~ character can be used when a neume appears without any syllable below it. The command will draw an underline under the relevant neume. Leaving a syllable empty will produce the same result. An empty neume block or the ~ character can also be used to produce a syllable with no neume above it (in this case, the neume block is left blank). To produce a neume with empty space (and no underline) below it, use any other command that is rendered as a space (for example, `\thinspace`). The following examples demonstrate the functionality:

`\cuKruk{𜽖}{}`	
`\cuKruk{𜽖}{~}`	
`\cuKruk{𜽖}{\thinspace}`	
`\cuKruk{~}{text}`	text

Note 3: fonts for Znamenny Notation usually provide color information for certain glyphs (such as the cinnabar marks) via data in the COLR and CPAL tables. However, COLR / CPAL technology is presently not supported in XeTeX or LuaTeX. When the churchslavonic package is loaded with the `autocolormarks` option, the `\cuKruk` and `\cuKrukPara` commands will automatically typeset the cinnabar marks in red by internally invoking the `\cuKinovar` macro. Additionally, if the churchslavonic package is loaded with the `gray` or `bw` options, the cinnabar marks will be typeset in grayscale or black, respectively.

Limitations: When a document is processed using XeTeX, attempting to render marks in color breaks normal OpenType glyph positioning rules. Loading the churchslavonic package with the `autocolormarks` option when XeTeX is used will produce a warning. Presently, you can only render marks in red and maintain correct positioning when using LuaTeX. When a document is processed using LuaTeX, the churchslavonic package by default loads with the `autocolormarks` option. If automatic coloring is not desired, it can be turned off in this case by loading the churchslavonic package with the `noautocolormarks` option.

The `\cuKruk` commands may be nested, which can be used to produce 'explanatory marks' («подобные пометы»):

```
\cuKruk{\cuKruk{\cuKinovar{\tiny 𜼀}}}
{\Large 𜽖}}{ла}
```

ла

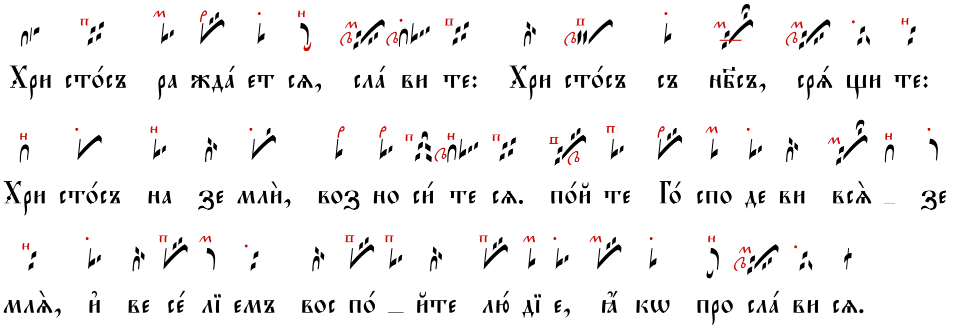
6.2 Controlling the Appearance of Znamenny Notation

A number of parameters may be used to control the positioning and appearance of Znamenny neumes:

`krukFont`	Specifies the font that is used to typeset the neumes
`sylSpace`	Specifies the amount of spacing around a syllable (default is `0.2em`)
`topMargin`	Specifies the amount of space (margin) above the neumes (default is `0.3em`)
`krukRaise`	Specifies the amount of space between the neumes and lyrics (default is `1em`)
`sylRuleHeight`	Specifies the thickness of the rule used when no syllable appears below a neume (default is `0.08em`)

The parameters are passed as options to the `\cuKruk` and `\cuKrukPara` commands, separated by commas. For example, we can modify the spacing of neumes in the above example:

```
\cuKrukPara[krukRaise=1.5em,topMargin=0.6em,sylRuleHeight=0.02em]{  ... \\
Хри-сто́съ ... }
```


Хри стóсъ ра ждá ет сѧ, слá ви те: Хри стóсъ съ нб҃съ, срѧ́ щи те:
Хри стóсъ на зе млѝ, воз но сѝ те сѧ. пóй те Гó спо де ви всѧ̀ _ зе
млѧ̀, и҆ ве се́ лї емъ вос пó _ йте лю́ дї е, ꙗ҆́ кѡ про слá ви сѧ.

References

[1] Aleksandr Andreev, Yuri Shardt, and Nikita Simmons. *Church Slavonic Typography in Unicode*, Uncode Technical Note 41. 2015. http://www.unicode.org/notes/tn41/