

Sorting in Hindi and Marathi, preliminary draft

Zdeněk Wagner, Anshuman Pandey, Jaya Saraswati

August 1, 2010

Abstract

This draft describes the first version of `xindy` modules for sorting in Hindi and Marathi. The document is intended for advanced users and developers. It is not currently ready for wide distribution, some problems are not yet resolved and the package is prepared for installation into T_EX Live. Installation to other T_EX distributions will require modifications in the Makefile's.

Contents

1	Introduction	2
2	Devanāgarī and L^AT_EX	2
3	Sort rules	2
3.1	Nasalisation	3
3.1.1	Orthographical note on nasalisation	3
3.2	Characters with <i>nuktas</i>	3
3.3	Conjuncts	4
3.4	Special characters	4
4	Sorting in transliterations	4
4.1	IITK transliteration	4
4.2	Velthuis transliteration	4
5	System description	5
5.1	Makefile	5
5.2	Directory zw-make-rules	5
5.3	Alphabets	5
5.4	Modules	5
5.5	Subdirectory transliteration	6
5.6	Subdirectory wordlists	6
5.7	Subdirectory latex	6
5.8	Directory doc	6

6	Problems	6
6.1	Sort rules documentation	6
6.2	Problem of अ	7
6.3	Problem of आ	7
6.4	Problem of <i>viramas</i>	7
6.5	Devanāgarī numeric location class	7
6.6	Page ranges	8
6.7	Missing Marathi support for babel	8
6.8	Editor problems	8
7	Conclusion	8
8	License	8
9	Links	9

1 Introduction

Indexes in L^AT_EX documents have been traditionally prepared by MakeIndex. It existed for English and German and a modified program, CsIndex, was developed for Czech and Slovak. Support for other languages was missing. It started to change with MakeIndex 3 which is now superseded by xindy.

Implementation of sort rules for a language on MakeIndex 2.x is difficult, a large part of the C program has to be rewritten. However, xindy makes use of tables that define the sort rules. Adding language support to xindy is thus relatively easy.

This preliminary system is based on xindy kernel version 2.3 as distributed with T_EX Live 2008 and xindy make rules version 0.2. The authors do not wish to upgrade xindy in the middle of development.

2 Devanāgarī and L^AT_EX

L^AT_EX was originally intended for languages using the Latin script. Devanāgarī was not supported directly. It had to be entered in transliteration and converted by a preprocessor to T_EX control sequences. X_LL^AT_EX can accept a source text directly in Unicode. Nowadays users, who need to write their texts in Devanāgarī, more often input the text in Unicode but the older system still remains in use. Since sort rules in xindy are implemented via tables, it is easy to prepare them both in Unicode as well as in one or more transliterations. This does not present any speciality, even languages using the Latin script may use several different encodings. Xindy is prepared to cope with such problems.

3 Sort rules

The sort rules in Indic languages almost copy the order of characters in Unicode but there are important changes. Vowels with *anusvara* and *candrabinu* precede vowels without

nasalisation. Word आँख therefore precedes आ. Hindi makes use of characters with *nukta* such as ज्ञ or ङ etc. Such characters share their alphabetic order with corresponding characters without *nukta*, i. e. ज and ङ. This can be handled by xindy tools.

3.1 Nasalisation

The perl code for defining the alphabet expects the group name, the lowercase character and one or more uppercase characters. Devanāgarī does not use uppercase characters but this property can be used for a different purpose. We define *candrabindu* as an uppercase variant of *anusvara*. Using this definition xindy will treat them as equivalent characters which is the correct sort order. We moreover ask xindy to place lowercase characters first so that पलंग will precede पलंग which are just two orthographical variants of the same word meaning *bed*. We can define the same group again with a different set of characters. For instance, the rules contain:

```
[ 'अ', [ 'अँ', 'अँ' ] ],
[ 'अ', [ 'अ' ] ],
```

Thus word अंत will precede अकती but will belong to the group bearing the अ heading, not the अँ heading.

3.1.1 Orthographical note on nasalisation

Nowaday's orthography requires nasal consonants preceding plosive consonants be written as *anusvara*. Correct spelling is thus खंड, चंद्र, कंबल but variants खण्ड, चन्द्र, कम्बल can frequently be seen. It is possible to handle these cases by defining merge rules that would convert ण्ड, न्द, म्ब etc. to a *nukta* followed by the respective consonant. This could be done both in UTF-8 and in a transliteration. However, such a merge rule does not solve all problems. If the index should contain an entry saying “चन्द्र, see चंद्र”, the merge rule would not allow it. Orthographical problems are therefore left to decision of authors and editors of a manuscript.

3.2 Characters with *nuktas*

Characters with *nuktas* are easily handled as if they were **\$ligatures**. Marathi does not use *nuktas*, thus the ligatures section is empty. Although Unicode contains characters with *nuktas* as glyphs, some authors write them as the base character followed by U+093C. This should first be normalized using the merge rules. This can be achieved by specifying `-M dvngnukta`. Notice that this problem occurs in UTF-8 only. These merge rules are useless if the input file is written in a transliteration. It must, however, be noted that Xe_{La}TeX can convert Velthuis encoding to UTF-8 on the fly by means of a TECKit map. Xe_{La}TeX will then see UTF-8 but this map will never write U+093C characters. If you do not combine such a text with characters entered directly in UTF-8, normalization by *dvngnukta.xdy* merge rules is not needed.

3.3 Conjuncts

Conjuncts do not present any problem at all. Exactly as the *fi* ligature in the Latin script, conjuncts in Indic scripts define just visual representation but are sorted according to individual characters from which they are composed. They are just two exceptions, namely **क्ष** and **ज्ञ** which are treated as special characters in some languages. **Xindy** tables have limited size but Indic languages can fit to them because conjuncts need no special handling.

3.4 Special characters

The list of special characters is extended with nukta (U+093C), zero-width-joiner (ZWJ), and zero-width-nonjoiner (ZWNJ). Nukta can appear in the words if normalization described in section 3.2 was not used. The other two characters may be used for fine control of ligatures generation but must not influence sort order.

4 Sorting in transliterations

All transliterations share common features. It would be error prone to try to define sort rules for each transliteration separately. Instead we define a master table in Unicode and transliteration tables defined in perl. **Xindy** tables are then generated automatically. We cannot use directly the table intended for Unicode. In Unicode, different codes are used for dependent and independent vowels but they are not distinguished in transliterations. Short dependent *a* must be written in transliterations although there is no such character in Unicode. On the other hand, *virama* is not used to build conjuncts in transliterations. Instead the consonants are just written adjacent each other without the short independent *a* vowel. The master table cannot be created automatically, development of such a program will take more time than hand editing. The master table makes use of the same features.

4.1 IITK transliteration

IITK is used by CFILT IIT, Mumbai. It has never been used with $\text{\TeX}$ and thus lacks any support in $\text{\TeX}$. The tables are defined just to show that any transliteration can be used with **xindy**. Sorted output cannot be used without other software tools.

4.2 Velthuis transliteration

Unlike IITK, Velthuis transliteration is not one-to-one encoding. For instance, **आ** can be entered as *aa* or *A*, **भ** can be entered as *bh* or *B*. Handling all these cases in the sort rules would be difficult. Fortunately **xindy** offers another tool. Such characters are first normalized by means of the merge rules. The very same mechanism is used to normalize the code generated by the INPUTENC package.

The sorted output requires preprocessing before it could be fed into $\text{\LaTeX}$. A few **xindy** modules are prepared for this purpose. They will be described in section 5.4.

5 System description

The following text is intended for the developers. It describes how the system is built. Advanced users must read it at least briefly, otherwise they will not be able to install it.

We provide not only the source files but also some generated files so that users can easily see the examples.

5.1 Makefile

The top-level makefile calls makefiles in all subdirectories. All of them have equal main targets. Target *help* gives you brief description of the targets. The help message is written inside the `makehelp.pl` script.

Important note: The system was tested under T_EX Live 2008 and 2009. The installation procedure is prepared for it. It might or might not be usable with newer releases of T_EX Live. The *install* target asks KPSEWHICH to expand the TEXMFMAIN variable in order to find installation directory. You must have write permission to install the modules.

5.2 Directory zw-make-rules

The `make-rules.pl` version 0.2 contained a bug that was reported and most probably is already fixed. However, as already written, it is not a wise idea to change a lot of files at the same moment. Instead of upgrading the authors keep their own copy but replacing it with the latest version should not cause problems.

5.3 Alphabets

The system defines alphabets for Hindi and Marathi in UTF-8 as well as in the Velthuis and IITK transliterations.

5.4 Modules

This subdirectory contains x^ondy modules. Subdirectory `inputenc` contains the module with merge rules for the Velthuis transliteration as mentioned in section 4.2. Subdirectory `base` contains modules for output in the Velthuis transliteration. Four types are available. Files **hindi** and **modernhindi** differ just in a preprocessor instruction for selecting a set of conjuncts. Files **-dn-** output the page number in Arabic while **-dnnum-** outputs the page number using Devanāgarī numerals. The directory contains also module *dvngnukta.xdy* defining the merge rules for normalization of characters with nuktas in UTF-8 as described in section 3.2.

5.5 Subdirectory transliteration

This directory contains the transliteration tables as well as the transliteration engine. Notice that transliterations are bound to scripts, not to languages. The transliteration engine `transliterate.pl` is documented via POD.

5.6 Subdirectory wordlists

This subdirectory contains hand-made word lists used for testing. Special words are often selected in order to test difficult cases. The word list is prepared in UTF-8 and transliterated to other encodings. Sorting an XML files by the Saxon XSLT processor is used just for comparison.

5.7 Subdirectory latex

This subdirectory contains $\text{\LaTeX}$ and $\text{\XeLaTeX}$ examples. The text in Hindi is taken from the Velthuis Devanāgarī package, the Marathi text was taken from Marathi Wikipedia. All words were marked for indexing irrespective of flexion. The intention was not to create high quality index, we just wanted to demonstrate index generation.

The `dvngnukta` directory contains a sample testing the merge rules for handling *nuktas* as described in section 3.2.

5.8 Directory doc

This directory should contain documentation. Velthuis Devanāgarī for $\text{\TeX}$ is used because it is portable across all platforms.

6 Problems

This section describes problems both in the build system and unsolved issues in sorting.

6.1 Sort rules documentation

The original make rules package documents the sort rules via $\text{\TeX}$. It is not straightforward for Indic scripts. If we tried to document them via transliteration, we would have to implement a complex transliteration system. If we decided to use $\text{\XeLaTeX}$ instead, another problem would arise. Although Devanāgarī fonts are freely downloadable, there is no such font that is guaranteed to be installed everywhere. Users of `xindy` would then have to install Devanāgarī fonts in order to process the documentation although they may not be interested in Indic scripts. Documentation of Hindi and Marathi sort rules was therefore abandoned at this stage.

6.2 Problem of ओं

Textbooks and dictionaries do not define the sort order of the ओं symbol. I have recently found that my Česko-hindský slovník / चेक - हिंदी शब्दकोश lists it as the ओम् word. Thus it seems that ओं is a symbol, not a character. It is not included in the alphabet but may be put into the index by other means if requested.

6.3 Problem of औ

औ is not an original character in Indic scripts and textbooks and dictionaries do not define its alphabetic position. It was introduced to transliterate words of English origin. However, words as औफिस, औयल etc. are widely used nowadays. We thus cannot ignore this character. Currently it is treated as if it were an uppercase variant of आ although we are not sure whether it is correct. Notice that in the Marathi Xe_{La}TeX sample word औफ appears between words आपल्य and आर्थिक.

6.4 Problem of *viramas*

In Hindi *virama* is not used to denote that a word ending with a consonant. However, there are still a few words of Sanskrit origin that retain terminating *virama*, e. g. महान् or वाक्. If Velthuis transliteration is used in @hindi or @modernhindi mode, words ending with a consonant are entered as they are pronounced, thus घर is written as *ghar*. If a terminating *virama* should be used, it must be written explicitly, i. e. महान् is entered as *mahaan_*. IITK has no character for *virama*. The short *a* vowel must be written even at the end of a word. Thus घर must be entered as *Gara* while महान् is written just as *mahAn*. The *virama* must always be written if the input is in Unicode. These differences make definition of the sort tables difficult. Moreover, Česko-hindský slovník / चेक - हिंदी शब्दकोश contains the following order:

महान् < महानगर < महानुभाव
वाक्रिया < वाक् < वाक्छल

These samples are in contradiction but at least the latter conforms to the Unicode codepoint order. In the current stage words ending with *viramas* are sorted incorrectly.

6.5 Devanāgarī numeric location class

Devanāgarī numbering is not easy to use as a location class. This is not a big problem in T_EX but presents limitation when used for building indices for other software tools. If Velthuis transliteration is used, numbers are entered as Arabic numbers. If \dnnum is used, the preprocessor will convert them to Devanāgarī numbers. The L^AT_EX counters may thus be defined as Arabic and switched to Devanāgarī by the \dnnum macro. Xindy will see Arabic numbers and will work with them. The rules supplied with this package will add \dnnum to the output, see section 5.4. Situation in Xe_{La}TeX is similar. Conversion from Arabic numbers to Devanāgarī can be achieved by the TE_Ckit map. Such a map

is distributed with Velthuis Devanāgarī for T_EX package. This approach is used in the X_YL^AT_EX Hindi example.

6.6 Page ranges

The L^AT_EX as well as X_YL^AT_EX samples make use of the standard page-range module. While it works in transliteration, the dash is displayed incorrectly in the X_YL^AT_EX examples. The problem is that the font used lacks this ligature. Page numbering in Devanāgarī is achieved by a TECKit map as described in section 6.5. There are two such maps available. We should use the *devanagarinumeralstex* map in the `\setmainfont` macro but in that case two dashes ask for a ligature not present in the font and an empty space between the numbers is displayed. The sample therefore makes use of the *devanagarinumerals* map where the ligature is not formed and two dashes remain visible. The problem can be solved by developing a modified module where the dash will be replaced either by a `\vrule` or by a dash from a visually compatible font. Briefly speaking, the ugly output is a problem of the font used, not a problem of x_Yndy.

6.7 Missing Marathi support for babel

Velthuis Devanāgarī for T_EX probably has not been used for Marathi so far. At least nobody has complained that Marathi support is missing, nobody has suggested the translations. While Hindi examples make use of the (not yet) official babel module in which `\indexname` is defined as सूची, the Marathi sample displays the index title in English.

6.8 Editor problems

I had to solve the problem when editing the perl scripts containing Devanāgarī characters. Some editors are not able to use non-Latin characters at all, some editors get confused by a mixture of Latin and Devanāgarī and after saving a file and reading it again it contains nothing but useless garbage. Finally I found that commercial <Oxygen/> XML editor is able to cope with it.

7 Conclusion

The software is provided as is, in the unfinished stage ready for testing. It is not guaranteed that it is usable for serious work. Authors will appreciate comments and suggestions. The main author may be reached at zdenek.wagner@gmail.com, the x_Yndy related list as well as Velthuis Devanāgarī list are open for their members only.

8 License

The license of the software is GPL V2+ in order to be compatible with x_Yndy itself.

9 Links

- A Flexible Indexing System xindy, <http://www.xindy.org/>
- Velthuis Devanāgarī for T_EX, <http://devnag.sarovar.org/>
- T_EX Live, <http://www.tug.org/texlive/>
- Resource Center for Indian Language Technology Solutions, Indian Institute of Technology Bombay, <http://www.cfilt.iitb.ac.in/>
- Author's web page, <http://icebearsoft.euweb.cz/>