Printing Transliterated Indian Language Documents

itrans

version 2.10

This is a project that aims to develop a single tool for handling the printing of various indian language documents, assuming that the input is in some transliterated form. The transliteration mechanism for all indian languages will be through the english language. And, only those languages that follow the basic structure as in the devanagari script will be included in this project. (Actually, most indian languages do follow similar structure—vowels, consonant, consonant-vowel forms, consonant-consonant ligatures, etc—Urdu being a major exception.)

This document describes the use of version 2.10 of the program *itrans*, which may be used to generate Devanagari or Tamil output from an english transliterated form. Two devanagari fonts are supported, one devanagari font is bundled in with this package, the other one is not bundled in but can be obtained from various anonymous FTP sites. The bundled in devanagari font is a PostScript font, called *devnac*. *devnac* was developed (and is under further development) by Avinash Chopde. The other font is a Metafont font called *devnag* and has been developed by Frans Velthuis. More details regarding *devnag* can be found in the file *dvng.itx*, more details regarding *devnac* can be found in *dvnc.itx*.

The tamil font is a Metafont font, and was created at the Humanities and Arts Computing Center of the University of Washington, USA. See the file *tamil.itx* for more information. The Metafont files for this font are also bundled with the itrans package. The tamil font is called *wntml*.

1 *itrans* Mechanism

itrans works by assigning every indian language letter an english equivalent. A transliteration table mapping devanagari characters into english equivalents is provided in the reference documents for the devanagari font: *dvnc.itx* for the Devnac font, and *dvng.itx* for the Devnag font. The tamil mapping is in the document *tamil.itx*.

itrans scans the input text for consonants, vowels, and special forms. Consonants suffixed with vowel codes create a complete composite character. Consonants may be suffixed with one or more consonants to create ligature forms. All this transliteration is automatically handled, and appropriate indian language characters are produced. If ligatures exist for a particular combination of consonants, they will be used automatically. If a ligature does not exist for some combination of the consonants, half-forms of the consonants will be used. The user can also override the ligature mechanism so that even if a ligature exists, the half forms of consonants will be used. Some languages such as tamil do not have any ligatures, in that case the appropriate action is taken — for tamil, a dot is printed on top of a consonant if it is not followed by a vowel form.

All these features make itrans an highly customizable and easy-to-use package. Even the transliteration map given here is not mandatory—the user can always edit the lexical source file and provide whatever mapping desired.

itrans is just a translation/char composition package. The task of actually placing characters on the page and spacing them correctly is left to other programs, such as TEX. TEX is the preferred interface, but, if absolutely necessary, and if the requirements of text placement are minimal, the dumb textual input interface could be used. This mode directly produces PostScript output, and contains minimal (read none) wordprocessing features; spaces and linebreaks in the input are copied to the output unchanged. This mode can only be used for fonts that are in PostScript itself, such as the devanagari font *devnac* that comes bundled with the system. Direct PostScript output cannot be produced for Metafont descriptions, such as the tamil wntml font.)

One important point to note is that since the current interface to this package is through an english language interface, it is quite clumsy and time consuming to start off writing a document or letter from scratch. I.e., thinking in marathi, and typing in english is quite slow! I find it much more easier and faster to write down a hand-written version of a document first, and then type it up in english, transliterating as I type. Of course, the problem could be attributed to my weak knowledge of marathi and hindi.

2 Input Format

itrans makes use of an IFM file — indian language font metric file, which is a ASCII file containing descriptions on how to generate the indian language characters from the basic characters available in the font.

Sidenote: This IFM file format is an *itrans* specific concept, it allows all character composition directives to be loaded in at runtime, making it easy to support many different indian languages. The IFM file is an ASCII file, and all the lines are in a format that the Adobe AFM file specification regards as a comment. Thus, in the future, when all PostScript RIPs really understand the composite character definitions, it *may* be possible to merge in the IFM file into the AFM file. See the technical reference manual, *tech.tex*, if you need more complete information regarding the IFM file format.

For example, the IFM file *dvnc.ifm* contains the descriptions on printing devanagari characters using the *devnac.ps* devanagari PostScript font, while *wntml.ifm* contains the descriptions necessary to print all tamil characters, based on the wntml.mf Metafont description.

itrans scans through the input text, and copies everything to the output unchanged, except for portions between marker words, such as `\marathi` and `\endmarathi`. Some eight–ten different marker words are available, see the Appendix A for more information. All english text between these words is mapped into indian language characters, based on the transliteration map.

At the beginning of the input file, the user has to specify the IFM file, and the name of the TeX or PostScript command that changes the font to the indian language font. For example, if the IFM file is named dvnc.ifm, and the font is available through the `\devnf` TeX command, the following two lines should be present in the input file:

```
\marathiifm=dvnc
\marathifont=\devnf
```

This also assumes the user will be using the markers `\marathi` and `\endmarathi`, see Appendix A for all the other language markers and commands.

Once the above initialization is made, the `\marathi` marker then specifies the beginning of the marathi transliterated text, and makes use of the specified IFM file (dvnc.ifm). At that point, *itrans* also outputs the font changing command (`\devnf`) specified in the `\marathifont` directive.

Note that both the TeX interface and the Dumb interface follow identical input text requirements. For further examples, see the sample documents provided. All TeX transliterated files have been given the file extension *.itx*. All Dumb Input transliterated files have been given the file extension *.ips*.

3 TeX Interface

itrans can accept TeX input and generate TeX output for all languages for which either a PostScript font description or a Metafont description is available. At the time of this writing, both devanagari and tamil are supported through this interface.

For example, the आ character in the table in the document *dvnc.itx* was produced by this input text:

`{\marathi aa \endmarathi}`

Thus, the character *aa*, when bracketed between `\marathi` and `\endmarathi` produces आ, when the appropriate IFM file name and font command name have been set correctly (as mentioned in the previous section).

For further examples, see the sample documents provided. All TeX transliterated files have been given the file extension *.itx*.

Normally, all english text between the words `\marathi` and `\endmarathi` is mapped into marathi characters, except for the word following a backslash character. I.e., you can include TeX commands in the transliterated text portion, the only restriction is that the command should be made up of letters and numerals only. Examples: `\indent or \hskip1in or \kern0.4em, etc`. Note that in the sample documents shown, TeX commands are usually surrounded by curly braces—they used as scope delimiters only, are not absolutely necessary, but are recommended.

3.1 TeX Requirements

To use this pre-processor, you must have the following tools:

1. TeX tools, including the capability to output dvi files using PostScript fonts (if you need to use the devnac devanagari font). This implies having *dvips*, release 5.41 or newer, a program written by Tomas Rokicki. If you don't have it, see the Frequently-Asked-Questions in the newsgroup comp.text.tex for details on how to get hold of *dvips*. Actually, if you only need to use tamil or Frans Velthuis's Devnag font, you don't need a dvi-to-ps converter. Those fonts are described in Metafont, thus any dvi-to-your-printer converter will be able to handle it (once you run Metafont and generate the printer specific files).

   For the IBM PC, the emTeX package includes the *dvips* executable, dvips for the PC also resides in the SIMTEL archives.

2. ANSI C compatible compiler, lex and yacc tools. This program is provided with the source code, and a makefile. The makefile is very specific to my machine, you will have to edit it to get it working correctly on your machine. If you do not have an ANSI C compatible compiler, you will have to make the "noansi" object, see the makefile for details; it automatically de-ansifies all the C sources. Also, if you do not have lex/yacc, modify the makefile to directly use the c-files provided.

4 Dumb Input Interface

Note: This section is applicable to the devanagari font devnac *only, the tamil font* wntml*, and the devanagari font* devnag *cannot be used for this purpose since they are not PostScript fonts.*

As mentioned earlier, the Dumb Input interface is an extremely primitive interface for printing, but will suffice for printing documents containing only devanagari text, and requiring no typesetting features such as centering, right flush text, etc. In addition to generating the devanagari characters, this method preserves the line breaks and spaces in the input text. So, unlike in the TeX version, which programatically decides where to break a line, here you have to include a end-of-line in the exact spot where you desire a new line to start in the output. And, if you need indented lines, you have to add spaces to simulate horizontal skips. Naturally, this approach may require several runs on the printer to get document to look correct.

The previous section titled *Input Format* applies to this interface, too. Thus, the user needs to specify the IFM file name, the font command name, etc before writing text between the indian language markers such as `\marathi` and `\endmarathi`. For further examples, see the sample documents provided, all dumb textual input transliterated files have been given the file extension *.ips*. The postscript prologue for itrans is in the file *itrans.pro*. Check the file out, it contains some useful postscript procedures.

4.1 Dumb Input Interface Requirements

1. All that is required is the capability to print PostScript files. Note that the *devnac* font used is a user defined PostScript font (Type III Font), so the printer (or the RIP used) must be capable of rendering Type III fonts.

5 Program Options

See the manual page for details regarding the program options need to be specified to itrans to make it run in the Dumb Input mode, and PostScript output mode. (As of this writing, the option was *-P*.)

6 Future Directions

This package has been developed with one goal in mind—to be able to handle transliterated documents for as many indian languages as possible. Many indian languages follow the concept of consonants, vowels, consonant-vowel pairs, ligatures, etc. (Pardon the non-technical terminology, my knowledge of indian alphabets is quite lacking!) This package has been developed to be flexible enough to allow all such languages to be used. In fact, for certain languages, such as hindi/gujarati, one could write a single transliterated document, and just change the keywords to produce text in a different language!

I am still looking for freeware fonts for these languages: bengali, gujarati, kannada, telugu, malayalam, or any other indian language. If you know of any source of freeware or public-domain indian language fonts for TEX or PostScript, please let me know (email: avinash@contex.com).

7 Known Problems

1. Version 2.10 *itrans* can handle simple comments in the transliterated text portion of the input file. % character begins a comment, the end-of-line terminates a comment. Handling of comments is not fully coded yet, and has some problems in that the % always starts a comment, only exception being when % is preceded by a backslash (\). Of course, there are many other instances where % should not begin a comment, those are ignored for now. (And may be ignored for ever, I think recognizing comments correctly would require too much effort.)

2. In the TEX interface, characters with any non-zero Y offset are not correctly printed in some cases, when the PostScript font Devnac is used. One such case is the da-ra ligature, words such as *draaviiDa* (द्रावीड). (Another case is the ha-u form हु, see how it is handled in the sample input file nehru.itx.) Note that it is only in certain cases that the word is printed incorrectly, in most cases it is handled correctly.

Usually, when the word appears near the end of the line, TEX (or dvips, but that sounds improbable) inserts a kern (a glue factor ?) just before the character with a non-zero Y offset, and the word appears squashed up or pulled apart at that point. Have no remedy for this, only workaround is to force a line break before the problem word, it usually sets everything right.

I have never encountered this problem when using Frans Velthuis's Devnag font with the itrans package, therefore this problem is probably related to the use of PostScript fonts in TEX. If you do notice this problem with the Devnag font too, please let me know.

8 Output Languages Supported

8.1 Devanagari Output

Two fonts are supported, *devnac* and *devnag*. *devnac* is a PostScript Type III font. This font can be used with both the TEX interface, and the direct PostScript interface (dumb textual) mode of *itrans*.

The font is named *dnh* in the TEX interface, and variations are also available, named *dnho*, *dnhrc*, *dnhre*. In the direct PostScript interface, the generic font changing commands *normalfont*, *slantfont*, *compressedfont*, *expandedfont*, etc have to be used. See the file itrans.pro to get a handle on the workings of the above PostScript commands.

This devanagari font tries to be a all-encompassing font, for the hindi, marathi, and sanskrit languages. If you feel it is missing some character or ligature, please let me know, send me e-mail at avinash@contex.com.

See the reference document *dvnc.itx* for the transliteration map and example texts. The IFM files for this font are *dvnc.ifm* and *mdvnc.ifm*.

devnag is a Metafont font, and thus can only be used with the TEX interface of *itrans*. This font is not bundled in with itrans, but has to be obtained from one of the anonymous FTP sites listed in the document *dvng.itx*. This font is supported by *itrans* in a limited manner, in that not all ligatures are available for use. This is because itrans can only handle two consonant ligatures, thus all the three or more consonant ligatures in *devnag* cannot be printed using *itrans*. Again, the document dvng.itx contains complete details and lists of the ligatures that are used and that are ignored by itrans. The IFM files for this font are *dvng.ifm* and *mdvng.ifm*.

8.2 Tamil Output

The single font *wntml* is currently supported. It is a Metafont font. This font can only be used with the TEX interface.

The Metafont descriptions of the font are provided, you can use them to generate the PK files of any font size desired. The Metafont commands for three sizes: 10, 12, and 17 point sizes are provided (wntml10.mf, wntml12.mf, and wntml17.mf, respectively).

This font was developed at University of Washington, and I would like to thank them for making the font available as freeware.

See the reference document *tamil.itx* for the transliteration map and example texts. The IFM file for this font is *wntml.ifm*.

9 Usage Hints

- As mentioned earlier, the system automatically uses ligatures whenever possible. For example, since the `ta-ta` ligature exists, input text of the form `tti` is printed as त्ति. If instead you need it to be printed as त्ति you have two choices. One, if you never want the `ta-ta` ligature to be used, you can edit the IFM file and comment out all the lines that refer to the `ta-ta` ligature. (The IFM file is a text file, for more information, see the technical documentation in tech.tex.)

  On the other hand, if you do want to keep the ligature, except in a few locations in the input text (say for small point type), or if you do not want to edit the IFM file, you can use the ligature inhibitors `{}` to prevent a ligature from being used. Whenever the `{}` characters are inserted between two consonants, *itrans* refrains from using the ligature (if it exists, if it does not, then these characters have basically no effect). Instead, the half-forms of the consonants (as appropriate) are used. Thus, even if the IFM file contains the `ta-ta` ligature, the input text `t{}ti` always appears as त्ति.

- Use the character pair {} to break the lexical scan. This becomes necessary because of the default behavior of the scanner, which tries to match the largest possible input pattern. Thus, when you write `nga` in the transliterated text, it comes out as ङ in marathi or ங in tamil. Now, that is just what you want for the `ng` consonant, but what if you wanted it to be scanned as two consonants: `n`, followed by `g` ? In such cases, you need to break the lexical scan, by introducing the

{} character pair to stop the scanner from associating `g` with the `n` preceding it. So, `n{}ga` in the input text results in: न्ग for marathi and ண்க for tamil.

This is a thing to watch out for in all cases where some consonant has a multiple letter mapping, and each letter by itself also represents some other consonant. In the above example, `ng` is the two letter consonant map, and both `n` and `g` represent other consonants.

- The fonts provided may be missing some or all punctuation characters, some may also be missing numbers. For example, the tamil font used, wntml, does not have any numerals or any punctuation characters. The devanagari font does have numbers and some punctuation marks (the double-quote is missing, for example).

  In all such cases, you can always get the required punctuation or digit character by ending the indian language transliteration scope (using one of the endmarkers), then printing the required punctuation mark, and the restarting the transliteration by using the start marker.

  An easier method, in TEX is to make use of the math mode for numbers. It is usually sufficient to use the $ enclosing scope to make numbers print correctly, since a $ enters mathmode and uses the math fonts. For punctuation marks, the user needs to explicitly change fonts: example:
  `\marathi......{\rm "}.......\endmarathi`.

  But that is preferable over this form:
  `\marathi......\endmarathi" \marathi.......\endmarathi`.

- How does itrans handle ligatures with more than two consonants ? For example, `shhTmii` contains three consonants. This ligature produces ष्ट्मी, the way itrans works is as follows: Beginning with the first consonant in the list, itrans checks if a double-consonant ligature has been defined for that consonant and the next one in the list. If such a character exists, then it is used and both consonants are consumed, and itrans repeats the procedure for the next consonant.

  There is one exception to the above rule: if at all possible, the last two consonants are handled together, that is if a ligature of the last two consonants exists, that is used over the pairing that would result from the above method. Example: `shhTrii` produces ष्ट्री, both `shha-Ta` and `Ta-ra` ligatures exist, but since the consonants `Ta` and `ra` are the last two consonants, that ligature is used over `shha-Ta`.

Of course, this default behavior can be changed by appropriately placing the ligature inhibitor sequence, `{}`. Example: `shhTr{}ii` produces ष्ट्री.

10 Author

This package has been developed by Avinash Chopde. Please e-mail comments to: avinash@contex.com. (postal address: 8 Kenmar Drive #55, Billerica, MA 01821. Phone: 508-663-0471.)

A Appendix: Markers and Marker Commands

You can use any of these marker sets to delimit the indian language text. The marker names indian/marathi/hindi/tamil do not actually do anything by themselves, but make use of the corresponding command names to load in the IFM file or output the font changing command string. So, use any one of the sets you feel suits your needs best.

1. `indian` marker.

   To set the IFM file name: `\indianifm=XXX.ifm`

   To set the font command name: `\indianfont=YYY`

   Start Marker: `\indian`

   End Marker: `\endindian`

2. `hindi` marker.

   To set the IFM file name: `\hindiifm=XXX.ifm`

   To set the font command name: `\hindifont=YYY`

   Start Marker: `\hindi`

   End Marker: `\endhindi`

3. `marathi` marker.

   (follow above examples, replace `indian` with `marathi`.)

4. `sanskrit` marker.

   (follow above examples, replace `indian` with `sanskrit`.)

5. `tamil` marker.

   (follow above examples, replace `indian` with `tamil`.)