TUG*India*

JOURNAL

INDIAN TEX users group

JAN. 1998
Vol. I No. 1

Indian TEX Users Group

The following is the Executive Commitee of the Indian TEX Users Group for the year 1998:

TUG*India* Journal is the quarterly publication of Indian TEX Users Group. The jounrnal will be distributed to all its members in electronic form (pdf format with interactive facilities) once in every three months.

Published by the Secretary, Indian TEX Users Group, TC 24/548, KRIPA, Sastha Gardens, Thycaud, Trivandrum, India, for and on behalf of Indian TEX Users Group. Typeset in LATEX 2ε and pdf generated with pdfTEX by River Valley Technologies, Software Technology Park, Trivandrum, India.

Contents

TUGIndia Journal **1** (1): 4–15 (1998)

IndicTEX/HTML, Traditional Scripts within Web-pages

Ross Moore
Mathematics Department, Macquarie University
Sydney, Australia 2109
e-mail: ross@mpce.mq.edu.au

Abstract: IndicTEX/HTML is a set of LATEX styles which interface to the various pre-processors which have been developed in recent years, for typesetting traditional Indic languages. Each LATEX package is accompanied by an implementation (in Perl) for use with the LATEX2HTML [1] translator. This simplifies greatly the process of creating multi-lingual Web-pages containing Indic scripts, from manuscripts written using TEX or LATEX and the fonts available therewith.

1 Introduction

Ultimately the best way to construct multi-lingual Web pages will be to use the Unicode[1] 16-bit font encoding scheme. However until appropriate 16-bit fonts become readily available and fully supported in the commonly-used browsers, other techniques are needed if one wants to present information using traditional Indic scripts, for example.

Alternatively, the HTML 4.0 recommendation[2] contains tags for specifying portions of a document intended to be presented using particular language conventions. Again browsers need to support these features, having suitable fonts available to display the required characters.

The approach adopted with LATEX2HTML [1] is more pragmatic. Multilingual Web pages are constructed using software that is freely available. By recognising that tools already exist for typesetting Indic scripts using TEX, and employing this software to generate images of individual letters, syllables, words, phrases, paragraphs or larger portions of text, LATEX2HTML constructs Web-pages that are fully compliant with current HTML standards.

The downside to this approach is the overall quantity of data that needs to be transferred, and the increased number of network accesses required to achieve this. Nevertheless the resulting HTML documents are created automatically from LATEX source and are viewable using existing browsers, with no need for extra “plug-ins”.

The original manuscript text, in whatever transcription/transliteration was used by its author, accompanies the HTML page as the value of the ALT attribute of <IMG> tags, provided this is not too long. With text-only browsers, this provides a view of the information. For larger portions, including images of whole paragraphs, the original transcription is included as a comment within the .html file.

[1] http://www.unicode.org/
[2] http://www.w3/org/TR/REC-html40/

2 Pre-processors, Fonts and Languages

The IndicTEX/HTML suite contains packages for LATEX 2ε to simplify the use of the fonts, macro-packages and pre-processors available at CTAN sites and mirrors; e.g. TUG's searchable CTAN site[3] This includes:

Indica Yannis Haralambous's[4] pre-processor, using fonts `sinha`, `sinhb`, `sinhc`, covers languages: Bengali, Gujarati, Gurmukhi, Hindi, Kannada, Malayalam, Oriya, Sanskrit, Sinhalese/Sinhala, Tamil, Telugu and Tibetan.

Available from CTAN:
`.../tex-archive/language/sinhala/.`

Devanagari for TEX [4] Frans Velthuis[5] uses the devnag pre-processor and dvng fonts to handle Indic languages, including Hindi, Marathi, Nepali and Sanskrit.

Available from CTAN:
`.../tex-archive/language/devanagari/.`

wntml — Tamil font using either pre-processor `tamilize` or `tmilize`, both from University of Washington, Tamil script can be processed.

Available from CTAN:
`.../tex-archive/language/tamil/.`

Malayalam-TEX [3] Jeroen Hellingman[6] uses pre-processors `mm` and `patc` to handle both traditional and reformed Malayalam script. The `mm` family of fonts allow for upright, slanted or bold faces in both normal and calligraphic styles.

Available from CTAN:
`.../tex-archive/language/malayalam/.`
The *patc* pre-processor is easily configurable, for both input and output. Malayalam-TEX comes with modules for Tamil (using the wntml font) and Devanagari (requiring also Velthuis's devnag pre-processor and fonts).

Note that these pre-processors, the fonts and any accompanying files of macros, are *not* distributed as part of IndicTEX/HTML. They must be obtained separately and may be subject to particular licensing requirements. Specific details on how to *use* these systems with LATEX2HTML is given in the following sections. One needs version v98.1, or later, of LATEX2HTML to be able to use these packages in the ways described here.

3 LATEX packages

Packages in the IndicTEX/HTML suite are named according to the languages or script supported. The extra package `indica.sty` is convenient, since Indica can support many languages simultaneously.

Filenames are restricted to at most 8 letters in the prefix, and are chosen to not conflict with names of other known packages. These are given in Table 3.
Each package has various options which can be specified with the `\usepackage` command. These options are used to specify the pre-processor and the particular transcription/transliteration scheme being used, when there is more than one possibility.

Using LATEX the packages have an effect only after the source has been pre-processed. However when using LATEX2HTML the source may be the compuscript *before* pre-processing. With many of the

[3] `http://ctan.tug.org/ctan/tex-archive/`
[4] `Yannis.Haralambous@univ-lillel.fr`
[5] `velthuis@rc.rug.nl`
[6] `etmjehe@genesis.etm.ericsson.se`

Language	LaTeX 2ε	LaTeX2HTML
Bengali	`bengali.sty`	`bengali.perl`
Devanagari	`devnagri.sty`	`devnagri.perl`
Gujarati	`gujarati.sty`	`gujarati.perl`
Gurmukhi	`gurmukhi.sty`	`gurmukhi.perl`
Hindi	`hindi.sty`	`hindi.perl`
Kannada	`kannada.sty`	`kannada.perl`
Malayalam	`malyalam.sty`	`malyalam.perl`
Oriya	`oriya.sty`	`oriya.perl`
Sanskrit	`sanskrit.sty`	`sanskrit.perl`
Sinhalese	`sinhlese.sty`	`sinhlese.perl`
Tamil	`tamil.sty`	`tamil.perl`
Telugu	`telugu.sty`	`telugu.perl`
Tibetan	`tibetan.sty`	`tibetan.perl`
Indica	`indica.sty`	`indica.perl`

Table 1: Languages and supporting packages

languages and options this is better, since it allows larger images to be produced. Otherwise one can get hundreds of images of individual letters or syllables. Furthermore this removes a step in the cycle of editing and testing.

In the following sections we look at the particular features of the packages, starting with indica.sty, since this covers most of the languages in a uniform way. For the Bengali, Gujarati, Gurmukhi, Kannada, Oriya, Sinhalese, Telugu and Tibetan languages, this is the only way they are currently supported in IndicTEX/HTML. Hindi and Sanskrit are supported by Indica, but these also can be used with devnag. Similarly Malayalam and Tamil are supported also by other pre-processors.

4 Indica

(An extended version of this section, with examples, is available for browsing)[7]

Yannis Haralambous' Indica pre-processor is a very flexible tool for processing transliterations and transcriptions of Indic languages. Producing TEX code to use fonts sinha, sinhb, sinhc it can provide a nice representation of almost all Indic scripts.

Languages supported explicitly are: Bengali, Gujarati, Gurmukhi, Hindi, Kannada, Malayalam, Oriya, Sanskrit, Sinhala or Sinhalese, Tamil, Telugu, and Tibetan. Other languages are implicitly supported insofar as their letters are shared with other scripts.

Furthermore the source document may be prepared in a variety of ways. Indica accepts the following forms [2] of input; combinations of these may be present within the same compuscript:

- a 7-bit (ISO–646) based on the Hindi/Sanskrit transcription by Frans Velthuis with some extensions, in particular for Sinhalese;
- the Classical Sanskrit Extended (CSX) encoding, an 8-bit extension of ISO–646 (cf. [5]), also with some further extensions;

[7] `http://www-math.mpce.mq.edu.au/texdev/Latex2 HTML/languages/indic/Indica/`

- LaTeX commands, building accented characters in accordance with the "standard" transliteration of Indic languages;

- Unicode (in the 16-bit part of ISO–10646–1), with a proposed Sinhalese encoding by Yannis Haralambous;

- Prasad Dharmasena[8] *samanala* transliteration scheme; for which support has been implemented within Indica by Vasantha Saparanadu[9].

4.4 Typesetting with LaTeX 2ε

The indica.sty package allows a compuscript, pre-processed using Indica, to be typeset with LaTeX 2ε. Loaded by: `\usepackage{indica}` .

This simply is an interface to load sinhala.sty, written by Dominik Wujastyk[10]

4.4 Making Web pages

When making Web pages, using LaTeX2`HTML`, the pre-processed compuscript can be used. However given the nature of the TeX source produced by Indica, this normally results in a single image for each syllable. One rapidly gets hundreds of images, especially if different font sizes are being used.

A better approach is to run LaTeX2`HTML` on the compuscript *before* preprocessing. For this an option must be given with the `{\usepackage}` command; in fact any option will cause the pre-processor to be called automatically when required to generate images of the traditional script. Now images will be generated of whole chunks of traditional script.

Each marked block of input, e.g. between the delimiters `#GUJARATI ... #NIL`, is recognised as needing to be rendered using images. Within such a block each paragraph becomes a separate image. The text of the original transcription/transliteration is included within the HTML document, as a comment (for each paragraph) which ought not show on-screen, but is nevertheless present in the `.html` file. Smaller inline chunks may not generate such a comment, but then the original source should be present as the value of the `ALT` attribute of the `<IMG>` tag for the image itself.

There are a large range of options which can be used, determining which preprocessor directives are to be included at the start of the LaTeX source to be used when images are generated. The available options are given in Table 2.

As many options as one likes can be selected with a single `\usepackage` command, or only a single one (to guarantee that the pre-processor is actually invoked). All pre-processor directives in the original source are included in the order in which they occur, so as to have the correct mode established when making a particular image.

Table 2 does not list all the options. For each supported language there are two 'alias' options; e.g. loading using `\usepackage[gujarati]{indica}`
generates the alias directive `#ALIAS GUJARATI G` , so that input for the Gujarati language may be delimited simply by `#G ... #N` , rather than using the longer `#GUJARATI ... #NIL`.

As names of some languages start with the same letter, a 3-letter alias is also supported as a package-option; e.g. `\usepackage[guj]{indica}` generates `#ALIAS GUJARATI GUJ` .

[8] `pk@isr.umd.edu`
[9] `vsaparam@laurel.ocs.mq.edu.au`
[10] `http://www.ucl.ac.uk/ ucgadkw/wujastyk.html`, which needs to be available on the local system. It can be obtained, along with the Indica pre-processor and fonts, from CTAN: `.../tex-archive/language/sinhala/`.

`\usepackage[preprocess]{indica}`	use the Indica preprocessor
`\usepackage[indica]{indica}`	same as `\usepackage[preprocess]{indica}`
`\usepackage[7bit]{indica}`	Frans Velthuis' 7-bit Hindi/Sanskri transcription
`\usepackage[csx]{indica}`	8-bit ISO–646 transcription with Haralambous' Sanskrit extensions
`\usepackage[latex]{indica}`	transcription using LaTeX macros of accented characters
`\usepackage[unicode]{indica}`	ISO–10646–1 (Unicode) with Haralambous' Sinhala extensions
`\usepackage[samanala]{indica}`	Prasad Dharmasena's *samanala* transliteration.
`\usepackage[bengali]{indica}`	`#ALIAS BENGALI B`
`\usepackage[tibetan]{indica}`	`#ALIAS TIBETAN T`
`\usepackage[ben]{indica}`	`#ALIAS BENGALI BEN`
`\usepackage[tib]{indica}`	`#ALIAS TIBETAN TIB`

Table 2: Package options for Indica modes

4.4 style considerations

For correct interpretation by LaTeX2HTML requirements on the compuscript are a little stricter than for TeX or LaTeX. However these are easily met with only minor changes to source that already typesets correctly.

Since LaTeX2HTML extracts from the source text those portions to be rendered as images, it must be able to unambiguously find the start and end. This is done using the markers `#GUJARATI ... #NIL` or `#G ... #N` , say.

However this extraction means that any other aspect that affects the typesetting for the image must be contain *entirely within* these delimiters. For example if the source is something like

```
... \bf ... #GUJARATI .... #NIL ....
```

then the extracted Gujarati script will *not* be set in bold-face. To have bold-face used in the image, simply arrange the source as:

```
... \bf ... #GUJARATHI \bf .... #NIL ....
```

5 Devanagari

(An extended version of this section, with examples, is available for browsing)[11]

When typesetting Devanagari script for inclusion in Web pages, one way is to the devnag pre-processor, written by Frans Velthuis. It builds TeX macros to typset syllables constructed using the dvng font, which is available in several sizes. This is sufficient to handle the Hindi, Marathi, Nepali and Sanskrit languages. The compuscript is pre-processed using the command:

```
devnag <infile>.dng <outfile>.tex
```

The `devnagri.sty` package provides an interface to the dev2e.sty package, written by Dominik Wujastyk, which handles the dvng fonts for LaTeX 2_ε. Alternatively one can use the original `dnmacs.tex` macros supplied with devnag, upon which dev2e.sty is based.

Loaded by: `\usepackage{devanagari}`.

[11] `http://www-math.mpce.mq.edu.au/texdev/Latex2HTML/languages/indic/Devanagari/`

Note that the devnag pre-processor and macro files are *not* provided as part of devnagri.sty, but must be collected separately from CTAN or elsewhere.

For LaTeX2HTML the package is implemented as `devnagri.perl`. This will work quite satisfactorily on the TeX code output by the devnag pre-processor.

It also works on the *input* compuscript. This is often more convenient, since the pre-processor is run automatically when required as a step in the conversion of document parts into images.

Loaded by: `\usepackage[devnag]{devnagri}`
or `\usepackage[preprocess]{devnagri}`.

5.5 patc pre-processor

Included with the Malayalam-TeX system, by Jeroen Hellingman, is a module to handle his transcription for Devanagari scripts. This module translates into a form suitable for the devnag pre-processor, which must then be used to complete the translation into TeX commands.

The `dng.pat` file provides the appropriate rules to convert Hellingman's transcription scheme[12] into that required for devnag.

```
patc -p <path>/dng.pat <infile> <outfile>.dng
```

Note that the `<path>` should give the full directory path to where the `.pat` pattern files are stored on the local system.

Source text pre-processed in this way can also be typeset to display the transcription form, using accented characters; e.g. having macrons and dots above and/or below letters, in places that do not occur with European languages. This requires the macro files `dntrmacs.sty` and `mmtrmacs` which both come with Malayalam-TeX.

With LaTeX2HTML, working on the original compuscript, both pre-processors are activated with the correct modes by specifying a single option with the `\usepackage` command.

Loaded by: `\usepackage[patc]{devnagri}`
or `\usepackage[hindi]{devnagri}`
or `\usepackage[marathi]{devnagri}`
or `\usepackage[nepali]{devnagri}`
or `\usepackage[sanskrit]{devnagri}`.

6 Tamil

(An extended version of this section, with examples, is available for browsing)[13]

6.6 Using the `wntml10` font

For typesetting Tamil script to create Web pages, one way is to use the `wntml10` created at the Univ of Washington, Humanities and Art Computing Center[14], in 1990.

Several pre-processors exist for converting various transcription schemes into TeX macros which use this font. The `tamil.sty` package is designed to work with files created using any of these pre-processors.

For `.dvi` output using LaTeX 2_ε, the package works with the *output* from the preprocessor. However with LaTeX2HTML the package, implemented as tamil.perl, can also be used with the *input* compuscript.

[12] `http://www-math.mpce.mq.edu.au/texdev/languages/indic/Mayalayalam/mmtrans`
[13] `http://www-math.mpce.mq.edu.au/texdev/Latex2HTML/languages/indic/Tamil/.`
[14] The Humanities and Arts Computing Center no longer exists. Although the Center for Advanced Research Technology in the Arts and Humanities (CARTAH) has a similar name, it does not appear to be involved in similar work.

This is often more convenient, since the pre-processor is run automatically, when required as a step in the conversion of document portions into images.

6.6 tamilize & tmilize, from University of Washington

These can be found at CTAN, in the same directory as the `wntml10` (The only difference is that the former uses the character `^^A` as an invisible 'escape' character, whereas the latter uses `\`.)
Loaded by: `\usepackage[tamilize]{tamil}` or `\usepackage[tmilize]{tamil}` .

6.6 patc, by Jeroen Hellingman

Written initially to handle Malayalam script, three different transcription schemes for Tamil are also supported. In what follows, `<path>` denotes the full directory path to where the `.pat` files are stored.

tamil Loaded by: `\usepackage[tamil]{tamil}`
the preprocessor command:

```
patc -p <path>/tamil.pat <infile> <outfile>.tex
```

converts from Hellingman's transcription into TEX macros. The transcription scheme is described in his manual pages.

adami Loaded by: `\usepackage[adami]{tamil}`
the preprocessor command:

```
patc -p <path>/adami.pat <infile> <outfile>.tml
```

converts text using the ADAMI transcription scheme into Hellingman's scheme.

wntml Loaded by: `\usepackage[wntml]{tamil}`
the preprocessor command:

```
patc -p <path>/wntml.pat <infile> <outfile>.tml
```

converts text using the transcription scheme developed at University of Washington into Hellingman's scheme.

Note that with **adami** and **wntml** the preprocessor must be run twice—once to convert to **tamil**, then again to get TEX macros. Both runs are performed automatically when LATEX2`HTML` is run on the original manuscript, with the correct option.

6.6 Indica

The other way to produce Tamil script is to use Yannis Haralambous' Indica pre-processor as discussed previously. This is an extremely flexible tool, capable of accepting various input modes and even different languages within the same source document, provided that each part is appropriately marked.

If only a single input mode is to be used, then the appropriate pre-processor directive can be included automatically by loading the package with option(s), as in Table 3.

The latter two methods of loading automatically create alias directives. Portions of text to be represented using traditional script can then be delimited[15] by `#T` ... `#N` or `#TAM` ... `#N` respectively, rather than using `#TAMIL` ... `#NIL`. The same results are achieved by loading `indica.perl` for Tamil:
`\usepackage[tamil]{indica}`
`\usepackage[tam]{indica}` .

[15] The pre-processor directive `#ALIAS NIL N` is always included automatically.

`\usepackage[7bit]{tamil}`	Frans Velthuis' 7-bit Hindi/Sanskri transcription
`\usepackage[csx]{tamil}`	8-bit ISO–646 transcription with Haralambous' Sanskrit extensions
`\usepackage[latex]{tamil}`	transcription using LaTeX macros of accented characters
`\usepackage[unicode]{tamil}`	ISO–10646–1 (Unicode) with Haralambous' Sinhala extensions
`usepackage[samanala]{tamil}`	Prasad Dharmasena's *samanala* transliteration.
`\usepackage[indica]{tamil}`	`#ALIAS TAMIL T`
`\usepackage[tam]{tamil}`	`#ALIAS TAMIL TAM`

Table 3: Indica modes for Tamil

Multiple options can be specified simultaneously; e.g. `\usepackage[7bit,samanala,tam]{tamil}` puts the following set of pre-processor directives at the top of `images.pre`:

```
#SEVENBIT
#ALIAS NIL N
#ALIAS TAMIL T
#SEVENBIT
#SAMANALA
#ALIAS TAMIL TAM
```

The first two lines are always placed, so the option `7bit` is actually redundant.

7 Malayalam

(An extended version of this section, with examples, is available for browsing)[16]

7.7 Malayalam-TeX

When typesetting Malayalam script for inclusion in Web pages, one way is to use Jeroen Hellingman's Malayalam-TeX. A demonstration version[17] is available at CTAN. This system comes with two pre-processors patc and mm.

The `malyalam.sty` package is an interface to Malayalam-TeX for use with LaTeX 2ε. It works by loading Hellingman's macro files `mmmacs.tex` and `mmtrmacs.tex` to interpret the TeX macros generated by the `patc` and `mm` pre-processors.

Loaded by: `\usepackage{malyalam}` .

Note that these macro files are *not* provided as part of malyalam.sty, but must be collected separately from CTAN or elsewhere.

For LaTeX2`HTML` the package is implemented as `malyalam.perl`. This is designed to work on the *input* manuscript[18] as well as on the output from the pre-processors. This is often more convenient, since the pre-processor is run automatically when required as a step in the conversion of document parts into images.

Loaded by: `\usepackage[<options>]{malyalam}`.

[16] `http://www-math.mpce.mq.edu.au/texdev/Latex2HTML/languages/indic/Malayalam/.`

[17] This is complete except for `METAFONT`sources; instead a range of pre-compiled sizes is included for the main font.

[18] As yet there are no explicit examples of Web pages constructed this way, since I've been unable to successfully compile `mm` for DEC Ultrix 4.4. Examples of Malayalam script use source code already pre-processed, accompanying the Malayalam-TeX distribution. They have been modified only slightly to be more compatible with LaTeX, rather than Plain-TeX.

When used with LaTeX 2ε specifying options has no effect, but does no harm, since the compuscript needs to have been pre-processed already. However with LaTeX2HTML specifying `<options>` requests a pre-processor to be called automatically, as required as a step in the creation of images of those portions of the document displaying traditional script.

patc

is a general pre-processor for the conversion of text using transcription or transliteration schemes into other forms. It works by applying a set of simple replacement rules; there is no analysis of syllable construction.

The `mm.pat` file provides the appropriate rules to convert Hellingman's transcription scheme[19] into TeX macros for including accented characters; e.g. having macrons and dots above and/or below letters, in places that do not occur with European languages.

```
patc -p <path>/mm.pat <infile>.mm <outfile>.tex
```

Note that the `<path>` should give the full directory path to where the `.pat` pattern files are stored on the local system.

Other transcription schemes can be handled; for example commands

```
patc -p <path>/ack2mm.pat <infile> <outfile>
patc -p <path>/mm2ack.pat <infile> <outfile>
```

convert between Hellingman's transcription and the A.C.K. scheme, designed by Mathai Chundat for the `alt.culture.kerala` newsgroup.

mm

handles the interpretation of syllables within the transcription text. It's output contains TeX macros which control construction of the traditional script characters, complete with diacritic markings.

This pre-processor is also configurable. It may produce either traditional script or the modern 'reformed' script. Indeed it is possible to have both forms within the same manuscript; this requires mm to be run twice, each time configured differently.

Loaded by: `\usepackage[mm]{malyalam}`
the pre-processor command:

```
mm <infile>.mm <outfile>.tex
```

translates to traditional Malayalam script, provided files `mm.scr` and `mm.trs` can be found in the current directory or on the path given by `$MMDIR`.
Chunks of text to be translated are delimited by: `<malayalam>` ... `</malayalam>` or `<ml>` ... `</ml>` or `$` ... `$`.

With LaTeX2HTML the pre-processor is used in various ways. In what follows `<path>` denotes the full directory path to where the `.pat` files are stored. It is deduced from the environment variable `$MMDIR` if this is set, else from `$PRE_FILTERS`. Either of these variables may be specified in `latex2html.config` or within a `.latex2html-init` file.

ml

Loaded by: `\usepackage[ml]{malyalam}`
the pre-processor command:

```
mm -s <path>/mm.scr -t <path>/mm.trs <in> <out>
```

translates to traditional Malayalam script.
Chunks of text to be translated are delimited by: `<malayalam>` ... `</malayalam>` or `<ml>` ... `</ml>` or `$` ... `$`.

[19] `http://www-math.mpce.mq.edu.au/texdev/languages/indic/indic/Malayalam/{}mmtrans`

mlr Loaded by: `\usepackage[ml]{malayalam}`
the pre-processor command:

```
mm -s <path>/mmr.scr -t <path>/mmr.trs <in> <out>
```

translates to reformed Malayalam script.
Chunks of text to be translated are delimited by: `<malayalam>` ... `</malayalam>` or `<mlr>` ... `</mlr>` or `$` ... `$`.

ack Load by: `\usepackage[ack,..]{malayalam}`
the pre-processor command:

```
patc -p <path>/ack2mm.pat <infile> <outfile>.mm
```

translates from A.C.K. transcription into Hellingman's, before making images. This option should be followed by another, either `mm`, `ml` or `mlr`. Chunks of text to be translated are delimited by: `$` ... `$` or `<malayalam>` ... `</malayalam>`.

Multiple options may be loaded with a single `\usepackage` command. Indeed packages for several Indic languages, perhaps using different pre-processors or the same pre-processor using different modes. (For example, Hellingman's patc has modes for processing Tamil and Devanagari scripts.)

In such cases the corresponding pre-processor commands are executed in the order of the requested packages and their respective options. This is important, to determine what happens to text-chunks delimited by characters common to two (or more) pre-processor modes; e.g. `$` ... `$` can be used with either traditional or reformed Malayalam script, or with a completely different pre-processor and script.

7.7 Indica

The other way to produce Malayalam script is to use Yannis Haralambous' Indica pre-processor as discussed previously. This extremely flexible tool is capable of accepting various input modes and even different languages within the same source document, provided each part is appropriately marked.

If only a single input mode is to be used, then the appropriate pre-processor directive can be included automatically by loading the package with option(s). The available options are as in Table 3, but with '`tamil`' replaced by '`malayalam`'.

`\usepackage[7bit]{malayalam}`	Frans Velthuis' 7-bit Hindi/Sanskri transcription
`\usepackage[csx]{malayalam}`	8-bit ISO–646 transcription with Sanskrit extensions
`\usepackage[latex]{malayalam}`	transcription using LaTeX macros of accented characters
`\usepackage[unicode]{malayalam}`	ISO–10646–1 (Unicode) with Haralambous' Sinhala extensions
`\usepackage[samanala]{malayalam}`	Prasad Dharmasena's *samanala* transliteration.
`\usepackage[indica]{malayalam}`	`#ALIAS MALAYALAM M`
`\usepackage[mal]{malayalam}`	`#ALIAS MALAYALAM MAL`

Table 4: Indica modes for Malayalam

The latter two methods of loading automatically create alias directives. Portions of text to be represented using traditional script can then be delimited[20] by `#M` ... `#N` or `#MAL` ... `#N` respectively, rather than using `#MALAYALAM` ... `#NIL`. The same results are achieved by loading `indica.perl` for Malayalam:

[20] The pre-processor directive `#ALIAS NIL N` is always included automatically.

`\usepackage[malayalam]{indica}`
`\usepackage[mal]{indica}`.

Multiple options can be specified simultaneously; e.g. `\usepackage[7bit,samanala,mal]{malayalam}` puts the following set of pre-processor directives at the top of `images.pre`:

```
#SEVENBIT
#ALIAS NIL N
#ALIAS MALAYALAM M
#SEVENBIT
#SAMANALA
#ALIAS MALAYALAM MAL
```

The first two lines are always placed, so the option `7bit` is actually redundant.

8 Future Extensions

Other specialised packages are available at CTAN archives, which are not yet supported within IndicTEX/HTML. This includes systems for Bengali, Gurmukhi, Sanskrit, Telugu and Tibetan. Current lack of support is in no sense a judgement by the author on the value of these systems. It simply is a consequence of the fact that there has not yet been sufficient time to install and test all these packages, identifing the various modes and any specific requirements for LaTeX2HTML, then produce appropriate interfaces.

LaTeXe's (and LaTeX2HTML's) package–option mechanism makes it very easy to extend systems such as IndicTEX/HTML, allowing a unified interface to whatever resources become available. It is to be expected that future versions of IndicTEX/HTML and LaTeX2HTML will include:

- support for packages at CTAN which currently are not supported;
- options to use the Ω system[21], in the form of its LaTeX adaptation LAMBDA, as the typesetting engine for images;
- an option to use Ω (or LAMBDA) to generate Unicode[22] output for the HTML pages, rather than images, starting from input using various transcription or transliteration schemes.

It accepts input using either [2]:

- a 7-bit (ISO–646) based on Frans Velthuis' Hindi/Sanskri transcription with some extensions, in particular for Sinhalese;
- the Classical Sanskrit Extended (CSX) encoding, an 8-bit extension of ISO–646 (cf. [5]), also with some further extensions;
- LaTeX commands, building accented characters in accordance with the "standard" transliteration of Indic languages;
- Unicode (in the 16-bit part of ISO–10646–1), with a proposed Sinhalese encoding by Yannis Haralambous;
- Dharmasena's `samanala` transliteration scheme; for which support has been implemented within Indica by Vasantha.

[21] developed by John Plaice and Yannis Haralambous
[22] `http://www.unicode.org/`

9 Bibliography

[1] Drakos N. & Moore R. R., *The LaTeX2HTML Translator* [manual accompanying software], online version at `http://www-math.mpce.mq.edu.au/texdev/Latex2HTML/dosc/manual/`, 1997.

[2] Haralambous Y., *A Sinhalese TeX System* [documentation accompanying the Indica pre-processor], available at `http://ctan.tug/org/ctan/tex-archive/language/sinhala/`, 1994.

[3] Hellingman J., *Malayalam-TeX, v1.1 User's Guide*, available at `http://ctan.tug/org/ctan/tex-archive/language/sinhala/`, 1994.

[4] Velthuis F. J., *Devanagari for TeX*, [manual accompanying devnag pre-processor], available at `http://ctan.tug/org/ctan/tex-archive/language/devanagari/`, 1991.

[5] Wujastyk D., *Standardization of Romanized Sanskrit for Electronic Data Transfer and Screen Representation*, [results of a session held at the 8th World Sanskrit Conference, Vienna, 1990] in *Sesame Bulletin* 4(1), 1991, pp. 27–29.

TUGIndia Journal **1** (1): 16–17 (1998)

Malayalam TEX

K. S. S. Nambooripad
Department of Mathematics, University of Kerala
Kariavattom, Trivandrum, India 695581
email: kssn@md2.vsnl.net.in

As mentioned elsewhere in this journal, the Indian TEX Users' Group has been constituted. An important objective of this organization is the development of TEX in the regional languages in India and to provide encouragement and support for its use as widely as possible. Accordingly, TUG India has started a project to create a workable TEX system in Malayalam; we hope to start similar projects in other regional languages as early as possible.

TEX is a typesetting system developed by Donald Knuth of Stanford University, U. S. A. It can create beautiful documents whose typographic quality is comparable to that of the world's finest printers. Later, Leslie Lamport created the high level macro package LATEX which made it possible for the authors to concentrate on the contents of the document rather than its form. Moreover, TEX is a free software which can be used on most computers from PCs to mainframes. The fact that its source code is freely available has encouraged many enterprising programmers to create various useful extensions to TEX and LATEX. As a consequence of this, TEX has become a typesetting system which is particularly suited for creating complex documents containing a lot of graphic insertions, tables, mathematics, etc. Its rich macro language can be used to automate many difficult and time-consuming tasks associated with production of documents. TEX has, therefore, become the default standard for typesetting technical documents and most of the leading publishers of technical journals and books are using it extensively.

Since TEX can run on PCs and other small computers, it is possible to create input files (which are pure ASCII files) of documents by small companies with a few PCs at their disposal or even by authors themselves on their personal computers. They can then be compiled and printed on inexpensive medium resolution ink-jet or laser printers which will be adequate for most purposes. If high quality output is desired, the same input files can be taken to photo-typesetters and get output of highest possible quality. This makes TEX particularly relevant in the Indian context, especially for publishing in regional languages. Thus availability of a workable TEX system in regional languages will facilitate publication (including electronic publishing) of high quality books even by small and medium publishers. Also it will facilitate production of quality documents in technical subjects such as engineering, mathematics, etc,. in regional languages for which facilities, existing at present, are quite inadequate.

Jeroen Hellingman has done pioneering work for Malayalam language by creating a preliminary TEX program for Malayalam which is quite usable. Similar system have already been created in other some Indian languages such as Sanskrit, Tamil, Kannada, etc. Meanwhile John Plaice and Yannis Haralambus have created a unicode implementation of TEX system called Omega. For several reasons, including the existence of large number of letters, conjuncts, etc., Omega system has several advantages over conventional TEX systems for regional languages in India (see the article by Yannis Haralambus about Omega systems elsewhere in this journal).[1] For example, with Omega system for Malayalam, it will be as easy to produce a document using traditional Malayalam script as to produce the same document

[1] would be published in the next issue

using new script; that is traditional Malyalam script will be an easy option for producing documents in Malyalam. Yannis Haralambus has kindly offered to create Omega system for Malayalam. We hope to give a sample output of his Omega system for Malayalam elsewhere in this journal.

What Haralambus will produce will only be a preliminary system. In order to create an efficient system capable of typesetting different types of documents, it will be necessary to do great deal of additional work. Initially only limited variety of fonts will be available; it will be necessary to produce a large number of additional fonts in various sizes and styles. An efficient hyphenation algorithm has to be developed. Various auxiliary programs for creating indexes, bibliographies, etc have to be created, etc. TUG India propose to organize a project to carry out these works. Jeroen Hellingman and Yannis Haralambus have offered their support and cooperation for the project.

TUG India being a voluntary organization, it needs material and intellectual support for successful completion of the project. We appeal to all philanthropic organizations, government agencies, universities and individuals to contribute liberally toward successful completion of this project which will give significant contribution towards improving the standard of publications in Malayalam.

TUGIndia Journal **1** (1): 18–24 (1998)

Hypertext marks in LaTeX: the hyperref package

Sebastian Rahtz
IT Department, Elsevier Science Ltd
The Boulevard, Langford Lane, Oxford OX5 1GB, UK
email: `s.rahtz@elsevier.co.uk`

Abstract: This paper describes release 6 of the 'hyperref' package, which provides a generalised interface to drivers or TeX variants which support hypertext features, including those which generate PDF.

1 Introduction

The package derives from, and builds on, the work of the HyperTeX project, described at `http://xxx.lanl.gov/hypertex/`. It extends the functionality of all the LaTeX cross-referencing commands (including the table of contents, bibliographies etc) to produce `\special` commands which a driver can turn into hypertext links; it also provides new commands to allow the user to write *ad hoc* hypertext links, including those to external documents and URLs.

The HyperTeX specification[1] says that conformant viewers/translators must recognize the following set of `\special` constructs:

```
      href: html:<a href = "href_string">
      name: html:<a name = "name_string">
       end: html:</a>
     image: html:<img src = "href_string">
 base_name: html:<base href = "href_string">
```

The *href*, *name* and *end* commands are used to do the basic hypertext operations of establishing links between sections of documents. The *image* command is intended (as with current HTML viewers) to place an image of arbitrary graphical format on the page in the current location. The *base_name* command is be used to communicate to the DVI viewer the full (URL) location of the current document so that files specified by relative URL's may be retrieved correctly.

The *href* and *name* commands must be paired with an *end* command later in the TeX file — the TeX commands between the two ends of a pair form an *anchor* in the document. In the case of an *href* command, the *anchor* is to be highlighted in the *dvi* viewer, and when clicked on will cause the scene to shift to the destination specified by *href_string*. The *anchor* associated with a name command represents a possible location to which other hypertext links may refer, either as local references (of the form `href="#name_string"` with the *name_string* identical to the one in the name command) or as part of a URL (of the form *URL#name_string*). Here *href_string* is a valid URL or local identifier, while

[1] This is borrowed from an article by Arthur Smith.

name_string could be any string at all: the only caveat is that '|"|' characters should be escaped with a backslash (\), and if it looks like a URL name it may cause problems.

However, the drivers intended to produce *only* PDF use literal PostScript or PDF `\special` commands. The commands are defined in configuration files for different drivers, selected by package options; at present, the following drivers are supported:

hypertex	dvi processors conforming to the HyperTEX guidelines (i.e. xdvi, dvips (with the `-z` option) and OzTeX)
dvips	produces `\special` commands tailored for dvips
dvipsone	produces `\special` commands tailored for dvipsone
ps2pdf	a special case of output suitable for processing by Ghostscript's PDF writer; this is basically the same as that for dvips, but a few variations remain
pdftex	Han The Thanh's TEX variant which writes PDF directly
dvipdf	Sergey Lesenko's dvi to PDF driver
dviwindo	Y&Y's Windows previewer

Output from dvips or dvipsone must be processed using Acrobat Distiller to obtain a PDF file. The result is generally prefererable to that produced by using the 'hypertex' driver, and then processing with `dvips -z`, but the dvi file is not portable.

2 Implicit behaviour

This package can be used with more or less any normal LATEX document by specifying

```
\usepackage{hyperref}
```

in the document preamble. Make sure it comes *last* of your loaded packages, to give it a fighting chance of not being over-written, since its job is to redefine many LATEX commands. Hopefully you will find that all cross-references work correctly as hypertext. In addition, the hyperindex option (see below) attempts to make items in the index by hyperlinked back to the text, and the option backref inserts extra 'back' links into the bibliography for each entry. Other options control the appearance of links, and give extra control over PDF output.

3 Additional user macros

If you need to make references to URLs, or write explicit links, the following low-level user macros are provided:

`\href{`*URL*`}{`*text*`}`

The *text* is made a hyperlink to the *URL*; this must be a full URL (relative to the base URL, if that is defined). The special characters # and ~ do *not* need to be escaped in any way.

`\hyperbaseurl{`*URL*`}`

A base URL is established, which is prepended to other specified URLs, to make it easier to write portable documents.

`\hyperimage{`*image URL*`}`

The image referenced by the *URL* is inserted.

`\hyperdef{`*category*`}{`*name*`}`text

A target area of the document (the *text*) is marked, and given the name *category.name*

`\hyperref{`*URL*`}{`*category*`}{`*name*`}{`*text*`}`

text is made into a link to *URL#category.name*

`\hyperlink{`*name*`}{`*text*`}`

`\hypertarget{`*name*`}{`*text*`}`

A simple internal link is created with `\hypertarget`, with two parameters of an anchor *name*, and anchor *text*. `\hyperlink` has two arguments, the name of a hypertext object defined somewhere by `\hypertarget`, and the *text* which be used as the link on the page.

Note that in HTML parlance, the `\hyperlink` command inserts a notional # in front of each link, making it relative to the current document; `\href` expects a full URL.

4 Package options

All user-configurable aspects of hyperref are set using a single 'key=value' scheme (using the keyval package) with the key `Hyp`. The options can be set either in the optional argument to the `\usepackage` command, or using the `\hypersetup` macro. When the package is loaded, a file `hyperref.cfg` is read if it can be found, and this is a convenient place to set options on a site-wide basis.

As an example, the behaviour of a particular file could be controlled by:

- a site-wide `hyperref.cfg` setting up the look of links, adding backreferencing, and setting a PDF display default:

```
\hypersetup{backref,
    pdfpagemode=FullScreen,
    colorlinks=true,backref}
```

- A global option in the file, which is passed down to hyperref:

```
\documentclass[dvips]{article}
```

- File-specific options in the `\usepackage` commands, which *override* the ones set in `hyperref.cfg`:

```
\usepackage[pdftitle={A Perfect Day},colorlinks=false]{hyperref}
```

In the key descriptions that follow, many options do not need a value, as they default to the value `true` if used. These are the ones classed as 'boolean'. The values `true` and `false` can always be specified, however.

4.4 General options

Firstly, the options to specify general behaviour and page size.

`draft`	boolean	*false*	all hypertext options are turned off
`debug`	boolean	*false*	extra diagnostic messages are printed in the log file
`a4paper`	boolean	*true*	sets paper size to 210mm × 297mm
`a5paper`	boolean	*false*	sets paper size to 148mm × 210mm
`b5paper`	boolean	*false*	sets paper size to 176mm × 250mm
`letterpaper`	boolean	*false*	sets paper size to 8.5in × 11in
`legalpaper`	boolean	*false*	sets paper size to 8.5in × 14in
`executivepaper`	boolean	*false*	sets paper size to 7.25in × 10.5in

4.4 Configuration options

`raiselinks`	boolean	*true*	In the hypertex driver, the height of links is normally calculcated by the driver as simply the base line of contained text; this options forces `\special` commands to reflect the real height of the link (which could contain a graphic)
`breaklinks`	boolean	*false*	Allows link text to break across lines; since this cannot be accomodated in PDF, it is only set true by default if the pdftex driver is used. This makes links on multiple lines into different PDF links to the same target.
`pageanchor`	boolean	*true*	Determines whether every page is given an implicit anchor at the top left corner. If this is turned off, `\tableofcontents` will not contain hyperlinks.
`plainpages`	boolean	*false*	Forces page anchors to be named by the arabic form of the page number, rather than the formatted form.
`nesting`	boolean	*false*	Allows links to be nested; no drivers currently support this.

4.4 Backend drivers

If no driver is specified, the package defaults to loading the `hypertex` driver.

`pdftex`	boolean	Sets up hyperref for use with the pdftex program.
`dvipdf`	boolean	Sets up hyperref for use with the dvipdf driver.
`nativepdf`	boolean	an alias for dvips
`pdfmark`	boolean	an alias for dvips
`dvips`	boolean	Sets up hyperref for use with the dvips driver.
`hypertex`	boolean	Sets up hyperref for use with the HyperTEX-compliant drivers.
`dviwindo`	boolean	Sets up hyperref for use with the dviwindo Windows previewer.
`dvipsone`	boolean	Sets up hyperref for use with the dvipsone driver.
`latex2html`	boolean	Redefines a few macros for compatibility with latex2html.
`ps2pdf`	boolean	Redefines a few macros for compatibility with Ghostscript's PDF writer, otherwise identical to dvips

Note that if you use dviwindo, you may need to redefine the macro `\wwwbrowser` (the default is `c:\netscape\netscape`) to tell dviwindo what program to launch. Thus, users of Internet Explorer might add something like this to `hyperref.cfg`:

```
\renewcommand{wwwbrowser}{C:\string\Program\space
  Files\string\Plus!\string\Microsoft\space
  Internet\string\iexplore.exe}
```

4.4 Extension options

`extension`	text		Set the file extension (eg `dvi`) which will be appended to file links created if you use the xr package.
`hyperfigures`	boolean		
`backref`	boolean	*false*	Adds 'backlink' text to the end of each item in the bibliography, as a list of section numbers. This can only work properly *if* there is a blank line after each `\bibitem`.
`pagebackref`	boolean	*false*	Adds 'backlink' text to the end of each item in the bibliography, as a list of page numbers.
`hyperindex`	boolean	*false*	Makes the text of index entries into hyperlinks. Easily broken …
`colorlinks`	boolean	*false*	Colours the text of links and anchors. The colors chosen depend on the the type of link. At present the only types of link distinguished are citations, page references, URLs, local file references, and other links.
`linkcolor`	color	*red*	Color for normal internal links.
`anchorcolor`	color	*black*	Color for anchor text.
`citecolor`	color	*green*	Color for bibligraphical citations in text.
`urlcolor`	color	*cyan*	Color for linked URLs.
`filecolor`	color	*magenta*	Color for URLs which open local files.
`pagecolor`	color	*red*	Color for links to other pages.

Note that all color names must be defined before use, following the normal system of the standard LaTeX color package.

4.4 PDF-specific display options

`bookmarks`	boolean	*false*	A set of Acrobat bookmarks are written, in a manner similar to the table of contents, requiring two passes of LaTeX. Some post-processing of the bookmark file (file extension `.out`) may be needed to translate LaTeX codes, since bookmarks must be written in PDFEncoding. To aid this process, the `.out` file is not rewritten by LaTeX if it is edited to contain a line `\let\WriteBookmarks\relax`
`linkbordercolor`	RGB color	*1 0 0*	The color of the box around normal links
`urlbordercolor`	RGB color	*0 1 1*	The color of the box around links to URLs

`filebordercolor`	RGB color	*0 .5 .5*	The color of the box around links to files
`citebordercolor`	RGB color	*0 1 0*	The color of the box around citations
`pagebordercolor`	RGB color	*1 1 0*	The color of the box around links to pages
`pdfborder`		*0 0 1*	The style of box around links; defaults to a box with lines of 1pt thickness, but the colorlinks option resets it to produce no border.

Note that the color of link borders can be specified *only* as 3 numbers in the range 0..1, giving an RGB color. You cannot use colors defined in TEX.

4.4 PDF display and information options

`baseurl`	URL		Sets the base URL of the PDF document
`pdfpagemode`	text	*None*	Determines how the file is opening in Acrobat; the possibilies are `None`, `UseThumbs` (show thumbails), `UseOutlines` (show bookmarks), and `FullScreen`. If no mode if explicitly chosen, but the bookmarks option is set, `UseOutlines` is used.
`pdftitle`	text		Sets the document information Title field
`pdfauthor`	text		Sets the document information Author field
`pdfsubject`	text		Sets the document information Subject field
`pdfkeywords`	text		Sets the document information Keywords field
`pdfview`	text	*FitBH*	Sets the default PDF ‘view’ for each link
`pdfstartpage`	text	*1*	Determines on which page the PDF file is opened.
`pdfstartview`	text	*FitB*	Set the startup page view
`pdfpagescrop`	n n n n		Sets the default PDF crop box for pages. This should be a set of four numbers

5 Defining a new driver

A hyperref driver has to provide definitions for eight macros:

1. `\hyper@anchor`
2. `\hyper@link`
3. `\hyper@linkfile`
4. `\hyper@linkurl`
5. `\hyper@anchorstart`
6. `\hyper@anchorend`
7. `\hyper@linkstart`
8. `\hyper@linkend`

The draft option defines the macros as follows

```
\let\hyper@@anchor\@gobble
\gdef\hyper@link##1##2##3{##3}%
\def\hyper@linkurl##1##2{##1}%
```

```
\def\hyper@linkfile##1##2##3{##1}%
\let\hyper@anchorstart\@gobble
\let\hyper@anchorend\@empty
\let\hyper@linkstart\@gobbletwo
\let\hyper@linkend\@empty
```

6 History and acknowledgements

The original authors of hyperbasics.tex and hypertex.sty, from which this package descends, are Tanmoy Bhattacharya (tanmoy@qcd.lanl.gov) and Thorsten Ohl (thorsten.ohl@physik.th-darmstadt.de). hyperref started as a simple port of their work to LaTeX 2ε standards, but eventually I rewrote nearly everything, because I didn't understand a lot of the original, and was only interested in getting it to work with LaTeX. I would like to thank Arthur Smith, Tanmoy Bhattacharya, Mark Doyle, Paul Ginsparg, David Carlisle, T V Raman and Leslie Lamport for comments, requests, thoughts and code to get the package into its first useable state. Various other people are mentioned at the point in the source where I had to change the code in later versions because of problems they found.

Tanmoy found a great many of the bugs, and (even better) often provided fixes, which has made the package more robust. The days spent on RevTeX are entirely due to him! The investigations of Bill Moss (bmoss@math.clemson.edu) into the later versions including native PDF support uncovered a good many bugs, and his testing is appreciated. Hans Hagen (pragma@pi.net) provided a lot of insight into PDF.

Berthold Horn provided help, encouragement and sponsorship for the dvipsone and dviwindo drivers. Sergey Lesenko provided the changes needed for dvipdf, and Han The Thanh supplied all the information needed for pdftex. Patrick Daly kindly updated his natbib package to allow easy integration with hyperref. Michael Mehlich's hyper package (developed in parallel with hyperref) showed me solutions for some problems. Hopefully the two packages will combine one day.

Especial extra thanks to David Carlisle for the backref module, the ps2pdf and dviwindo support, frequent general rewrites of my bad code, and for working on changes to the xr package to suit hyperref.

TUGIndia Journal **1** (1): 25–35 (1998)

Book Design for TEX Users Part 1: Theory

Philip Taylor
The Computer Centre, Royal Holloway and Bedford New College
University of London, Egham Hill, Egham, Surrey TW20 OEX
United Kingdom
p.taylor@vax.rhbnc.ac.uk

Abstract: Book design cannot be taught; it can only be learned, preferably by critical study of as many books as possible. Of all the elements which make up a book, white space is frequently the least considered and the most important. *Avant garde* designs are compared and contrasted with more conservative and traditional approaches. Three key elements: *uniformity, information* and *structure* are identified, and 'good design practice' discussed in terms of each of these.

Keywords: Design, typography, layout

There can never be too little space below headings,
only too much!

1 Introduction

The widespread use of TEX and other typesetting or DTP packages by tens of thousands of scientists, researchers and other academics has resulted in two rather disturbing phenomena: (1) more and more people are spending ever longer trying to get their publications to *look* right, rather than worrying about whether such publications are factually correct or are well written, and (2) fewer and fewer people, on opening a book for the first time, think first about the content, but instead commence by judging the book on its form, or to be more precise, on the appearance of the design and typesetting. We are, in fact, becoming a generation of self-taught designers and typographers, but in so doing we are tacitly avoiding the many years of training, apprenticeship and indenture which previous generations have deemed necessary.

This is, in itself, no bad thing – there are far too many self-appointed 'experts' ever ready to initiate neophytes into the arcane mysteries of their craft, in exchange for not inconsiderable sums of money – but in order for learning by osmosis to be effective, the beginner has to be exposed both to good and to bad examples of the art, and to think critically about what it is that differentiates the former from the latter. In Departments of Typography and Design, such examples abound, and the professors daily compare and contrast good with bad to the great benefit of their students; but in the incestuous world of TEX, good examples are rare whilst bad examples abound.

But why should this be? What is it about TEX, which in skilled hands is capable of producing results equalling the very best examples of hot lead composition, that encourages the production of second- and even third-rate design? I suggest that there are two main answers to this: (1) in *The TEXbook*,

This is the first part of a talk delivered to a SOFSEM meeting in Hrdonov (Czech Rep.) and reprinted with permission from SOFSEM Organizing Committee, Masaryk University Brno, Czech Republic and the second part would appear in the next issue.

which is presumably the first (if not the only) book on typesetting that users of TEX[1] encounter, there is extraordinarily little guidance given concerning document design, as opposed to document formatting, and (2) the standard styles which accompany LATEX generate results which even the staunchest LATEX adherent would have difficulty arguing represent "the state of the art" in document design, and which if considered dispassionately might justifiably be said to lack subtlety and finesse.[2]

Thus the lack of explicit guidance, together with the rather poor examples generate by the standard LATEX styles, has resulted in a proliferation of poorly designed books all of which shriek "TEX" (or "LATEX").[3] It would not be fair on the authors to adduce particular examples of this creed of mediocrity, but a glance at any reasonably complete library of TEX-related (or even TEX-set) books will shew what I mean. . .

However, all is not lost: a new generation of TEX-setters are emerging who appear to have studied the typesetter's craft, and several of the more recent books on TEX shew every evidence of having been *designed*, rather than having been ripped untimely from their progenitor's womb. In this paper, then, I propose to discuss what it is that differentiates a well-designed book from one that is ill-designed (or worse, one that has not been designed at all); and in so doing, I hope that I can in some small way contribute to a more universal adoption of TEX. For all the while that professional publishing houses see only bad examples of TEX-set books, they are unlikely to consider adopting TEX as a house standard; but if the general standard of TEX-set books can be raised to a point where they are either indistinguishable from, or even better than, books produced by traditional means or by commercial typeseting packages, then simple economics will ensure that such publishing houses give TEX the consideration it so richly deserves.

2 The Book

We all know what a book is, for we handle them every day; assuming a Western culture, it is basically a set of uniformly sized sheets of paper, joined at their left edge in some way, sandwiched between two slightly larger sheets of a more rigid or robust material that wraps around the left edge. It is differentiated from a magazine primarily by virtue of its cover: that of a magazine is only slightly more robust (although frequently more glossy) than the pages which it protects, whilst that of a book is almost invariably either thicker, or more rigid, or both; a magazine cover, too, has only one basic point of articulation, whilst most book covers articulate independently of each other. One other aspect separates the all but the thinnest book from all but the thickest magazine: a book is typically bound in *signatures* whilst a magazine is usually stapled through its spine as a single entity.

But open a book and open a magazine, and we see that these differences are only superficial; for there are far more fundamental differences which manifest themselves once inside he cover. The magazine is characterised by variation – each page is clearly different from the preceding and the next; whilst the book is characterised by uniformity – each page, seen from a distance, is virtually indistinguishable from the next (special pages apart). And in this uniformity lies the basis of successful book design; for readers have come to *expect* this uniformity, and anything which detracts from it will serve only to distract the reader.

Yet uniformity of itself is not enough: we could achieve uniformity by leaving each page blank, or by simply placing a large black rectangle within the margins of each page; but this will not satisfy our reader, who looks not only for uniformity but for *information*. Indeed, information is the very *raison* d' être of a book: without it, the book serves no purpose at all, and is at best a work of art (and at worst is totally valueless).

---

[1] as opposed to LATEX

[2] The Dutch, always sensitive to such issues, have produced a substyle 'Sober' which attempts to tone down the worst excesses of the default LATEX styles.

[3] Knuth, in his closing exhortation, wrote: "GO FORTH now and create *masterpieces of the publishing art*." Nowwhere, so far as I can trace, did he write: "and let every one of them shriek 'TEX' from every page". . .

So the book exists to supply information; and anything which inhibits or interrupts the flow of information from book to reader will diminish its value. If the flow of information is too badly affected, the reader will simply cast the book aside (how many of us, on attempting to read a page of reversed-out Bodoni in some otherwise traditional magazine, have simple given up and left the material unread? I have, many times, and cursed the designer for his/her stupidity in putting form before function).

Uniformity, information: what else? Well, if the book is in any sense *technical* (by which I exclude the novel but include almost everything else), then it is also *structured* (indeed, as we shall see, even a novel is structured in many senses, but not in the one which I am using here); and, possessing structure, it is capable of being accessed in a structured manner. It will have, at the very least, a table of contents; it *should* have an index (although far too many books that would benefit enormously from an index are lacking in this respect), and it may also possess an internal structure, in that the reader may be asked from time to time to *see Chapter 3, or see also Section 2.4.2*.

And these three elements, I suggest, lie at the heart of successful design: uniformity, information, and structure. We will look at each in turn to see how it may best be achieved, implemented or accomplished.

3 Uniformity

Take a book (a traditional book, not one hot off the presses of the DTP revolution), and flick the pages, rather as if there were one of those old-fashioned animated cartoon characters lurking in the corner of each page. What do you see? Most people perceive a regular grey grid: not black and white – you only see that if you look at static page – but instead a grey blur where the text appears, and white where there is no text (or other material). What is significant is that the white appears at the same place on every page: above the headline, below the footline, between the head/footlines and the body of the text, and to left and right of the text, in the margins of the page. If the book is set in multiple columns (usually two, but rarely more, except for rather specialised works), then a further block (or blocks) of white space will appear, separating the columns from each other.

And in many senses, this white space is that most important of the graphic elements which will go to make up each page. It provides the framework or matrix within which the 'dark matter' – the text, graphics, etc., which make up the *information* content of the page – is set. But probably because it does not itself appear to carry any information, it is frequently afforded less respect than it deserves, particularly by those undertaking design without any formal background. And yet, although it does not *appear* to carry any information, in fact it carries a great deal: without it, we would not know where the headline stopped and the page body began; where the page body stopped, and the footline began; where the left column stopped and the right began, and so on. . . In fact, it is *vital* to our comprehension of the contents of the page, and is therefore *at least* as important as every other element on the page, if not more so.

Because the white space and the dark matter are inextricably interlinked – one starts wherever the other stops, until the physical limits of the page are reached – any discussion of the uniformity of white space must equally be interlinked with a discussion of the uniformity of the dark matter of the page. But there is a third element to this uniformity which is even more dependent upon the inextricable interlinking of white space and dark matter, and that is the sense of 'greyness' of every page. The human eye is remarkably sensitive to small variations in grey level, and if the apparent greyness varies either within a page or between pages (particularly between facing pages, which form a *spread*), the effect can be quite discomforting. Such variations in apparent greyness can result from a variety of causes, of which the most common are: (a) use of letterspacing for justification; (b) inconsistent leading between two or more blocks of text in the same font; (c) inappropriate changes of leading or font (or both) when deliberately setting a block of text in a different font (in a multiline quotation, for example). The cures for each of these ills are farily straightforward: *never* use letterspacing to achieve justification, unless its use is so subtle that the eye cannot perceive the variation in inter-letter spacing; *never* allow the typesetting system to vary the leading in order to achieve vertical jusification (and never set two blocks of text in the same

font but with a different leading without being aware of the effect which will be achieved); and be aware of the perceived grey-level (white-matter: dark-matter ratio) when setting blocks of text in different fonts.

In an ideal world, attention to the suggestions of the previous paragraph would do much to ensure that the apparent greyness of each page was uniform: but there is another problem which results from our less-than-perfect world which can also significantly affect perceived greyness, and this is the problem of 'print through'. Ideal paper presents a uniform opaque whiteness on which the black of the ink is superimposed; real paper, on the other hand, whilst uniformly white (at least, as far high-quality printing papers are concerned) is rather less than opaque; if held up to a bright light, even the best paper will allow some light to shine through, and poorer papers are so translucent that printed material can be read almost as easily from the back as from the front (albeit as a mirror image). This in itself would be no problem were it not for the fact that the two sides of each sheet are logically independent entities: not only are they printed in separate operations, but the material appearing on one side bears little or no correlation with the material appearing on the other. However, in *designing* such pages, the effect of their back-to-back nature must be borne in mind, and a good design will attempt to ensure that each line of text on the obverse is matched by another line of text on the reverse. Of course, in practice this is not achievable; sections break up the flow of the text, as do illustrations and other graphics: but it must be the *intention* of the designer to achieve this line-for-line equivalence, and on this philosophy is predicated the whole concept of the *grid*.

The grid represents an abstract model of each page; special pages (e.g. chapter openings) may be afforded a special grid of their own, but normal 'running' pages will each use the same grid, onto which is mapped the various elements of the page. The grid can be perceived in hierarchical manner: at its most superficial, it will have lines for the physical limits of the page, for the left and right limits of the text (or of the columns, if a multi-column work), for the upper and lower limits of the page body, and for the headline and footline. At the next level of refinement, the page body will be divided into lines of text (which is why most traditional specifications for books express the dimensions of the page body in terms of lines of text, rather than so many picas or so many inches or centimetres). Superimposed back-to-back, two of these grids will intermesh perfectly, each line of text on one side corresponding to a line of text on the other; the left edge of the text on the obverse will exactly align withe the right edge of the text on the reverse, and so on (which has implications for the margins, as we shall see).

Of course, the grid is an ideal, but conformity to the grid must at times be allowed to be violated; if this were not the case, there would only be a finite number of positions at which a heading (for example) could appear above the text which it introduces: one line, two lines, three lines, etc. But such granularity is far too coarse for the aesthetic demands of real book design, and headings therefore need to be treated as special cases, allowed to float away from their 'natural' grid line whilst the paragraphs of text above and below the heading remain bound to the grid. Illustrations and graphics, too, must be treated as special cases, and float within the white space equivalent to an integral number of lines of text, thereby themselves being independent of the grid whilst leaving their surrounding paragraphs locked firmly in place.

But sometimes the requirements of page makeup will dictate that a particular page be underfull: a paragraph, for example, may finish in such a way that there is insufficient room (e.g., only a single blank line) for a new paragraph to start; or there may be room for a heading but not for a heading plus post-heading vertical white space plus at least two lines of text. How, in those circumstances, can the contradictory requirements of uniformity and page makeup be reconciled? In the limiting case, there is no general solution which will always work, and practical (real-life) books may occasionally have to violate one or other constraint; but equally often there is a solution which is both elegant and aesthetically pleasing: violate, by the same amount, the constraint of uniformity for two facing pages (*i.e.*, for a *spread*). If, for example, the verso (left-hand) page runs one line short, then *force* the recto (right-hand) to run one line short also; if the verso page would ideally run one line long, then allow it so to do, but require the matching recto page to run one line long also.

And in this concept of balancing the *spread*, as opposed to achieving uniformity between *all* pages,

lies, I believe the essence of good design. For when the book is held open in the hand, or laid open on the desk or lectern, it is not a single page that is seen at all, but a double-page spread; and if the two facing pages of the spread appear uniform (uniform in greyness or visual density; uniform in placement of headline and footline; uniform in size of margins – outer margins the same size as each other, inner margins also the same size but not necessarily the same apparent[4] size as the outer; and uniform in terms of grid-lock, in that verso lines appear in perfect vertical alignment with their recto counterparts) and balanced (with both verso and recto page bodies starting at the same height from the bottom of the page, and extending for the same depth), then much will have been accomplished; and if this same uniformity and balance can be carried through every spead of the book (thereby avoiding problems of print-through and so on), then much of the framework of good design will already be in place.

But there are many practical problems associated with the concepts of gridlock and balanced spreads; some of these are particularly true when using TEX, whilst others are more general. Those that are particular to TEX will be addressed in the sequel to this paper[5], whilst those that are more general are discussed below.

Considering first the problems of balanced spreads: it was suggested above that if the natural height for a verso page was one line short, or one line long, then it should be set to its natural size and its counterpart rector page *forced* to the same size. But what if the verso page naturally sets at the target size of the page, whilst the recto page runs one line light or one line over? If page makeup is performed on a page-by-page basis, then it is already too late to re-set the verso page, and either the recto page will have to be set to a non-natural size (if there is sufficient flexibility in the page makeup to allow this), or the balance constraint for the spread violated. And therefore we must postulate that any typesetting system intended for the production of well-designed books *must* be capable, at the very least, of setting a *spread* as an entity, rather than a page. Of course, this does not let us off the hook completely: for example, if the verso page naturally runs one line over, but the recto page finishes a paragraph at the natural height for the page, then it may not be possible to graft an additional line onto the recto page without violating some other (tacit) constraint; in these circumstances it may be necessary to backtrack even further, and to start asking questions such as "what if I were to set the preceding spread one line light, or one line over", and so on; in the final analysis, the more decisions about page makeup that can be deferred, the better the final volume is likely to be. As computer memory becomes cheaper and cheaper, it is by no means unreasonable to think about optimising a complete chapter at a time.

And what of uniformity: what if a multiline quotation, set in a smaller font with correspondingly reduced leading, *must* appear as an entity on a page, whilst there is no matching quotation on the other (physical) side of the same page? Then print-through will undoubtedly occur for the duration of the quotation, and in the worst case there will be an interference effect as the lines of the quotation drift into and out of synchronism with the lines of paragraph text on the other side. Here no matter how much material we accumulate can a makeup solution be postulated; and in the end we are dependent more on the skills of the paper maker in achieving near opacity than we are on our own skills in using and programming our typesetting system.

But there is much more to uniformity than simply gridlock and balanced spreads. Uniformity is a concept which percolates every element of good book design. Consider, for example, the treatment of chapter headings, section headings, paragraphs, quotations and so on: in what sense can they, too, be made 'uniform'? Clearly each must be unique, in order for the reader to immediately identify at what sort of entity he or she is looking; yet if they are not only unique but are also afforded wildly disparate typographic treatment, than any sense of coherence is lost and the book starts to take on the appearance of a mismatched hotchpotch of design ideas.

We might start by positing that there should only be a small number of different fonts used – 'the fewer the better' is hard to equal as an axiom for the selection of fonts! – whilst equally there should

[4] I use the term *apparent* here quite intentionally, for as we shall see, the apparent size of the inner margins is always less than their actual size, by an amount which is a function both of the thickness of the book and of the binding technique (s) used.

[5] Book Design for TEXUsers; Part 2: Practice, elsewhere in this volume

only be a small number of placements. For example, if paragraphs are fully justified (as would usually be the case for a book, although exceptions to this rule will be discussed elsewhere) and if section headings are set ranged left, then the book as a whole should probably restrict itself to thse two styles of setting: it would normally be inappropriate to have centred headings in a book that otherwise has a fully-justified or ranged-left theme running through it. But if section headings are set ranged left (perhaps in conjunction with semantic line breaks[6]), whilst normal paragraphs are set fully justified, then quotations could either set fully justified (like paragraphs) or ranged left (like section headings), but should probably not be set ranged right without good reason.

And what of indentation? Here two different schools of thought obtain. One would argue that the requirement of uniformity encompasses indentation, and that the indentation, once chosen, should apply to the whole book: thus, for example, lists would be indented by the same amount as paragraphs; quotations might be set with an additional left margin equal to this indentation; and the bibliography might be set with reverse indentation also equal to this same amount. The other would say that the requirements of clarity and lack of ambiguity dictate that a *different* indentation should be used wherever different entities occur, thereby giving the reader maximum indication of the nature of the entity being indented even on the most superficial glance at the page. I have sympathy with both points of view, but my inherently conservative background renders the former more appealing than the latter; I do not think I have yet seen an example in which the reader could have been mislead had a uniform indentation been adopted. But this whole area transcends the boundary between *uniformity* (which suggests a uniform indentation), and *information* (which suggests different indentations for different purposes), and brings us naturally to the next section.

4 Information

The primary function of any book is to convey information; yet the preceding discussion has concentrated almost entirely on the aesthetics of book design, rather than on its rôle as a medium for the communication of information. However, provided that the two ideas do not come into conflict, a uniform and aesthetically pleasing appearance does much to assist the book in its communication rôle, for it allows the reader to concentrate on the text (*i.e.*, the *information content* of the book) whilst not being distracted by its design (a fact which is sadly ignored by many of today's more *avant garde* designers). But there comes a point at which excessive adherence to the precept of uniformity would start to detract from the book's primary rôle as information source, and it is therefore to this area that we must now turn our attention.

Consider first of all the rôle of section headers: those single (or occasionally multiple) lines of text which serve to introduce the reader to the ideas which follow. This paper, for example, makes use of only a single level of section header, the author preferring to lapse into straight prose within each section; other authors, particularly those with a strong scientific background, feel happier if thy can classify their ideas in a strongly hierarchical manner, and frequently have recourse not only to *A-heads* (as in this paper), but B-heads, C-heads, D-heads and even E-heads on rare occasions. The first requirement for such headers is that thy shall, *unambiguously*, refer to the text which follows: it should not be possible, in a well-designed book, to mentally attach them to the preceding text. The mean by which this is accomplished is simplicity itself, yet is so often violated in amateur-designed books and other documents that one wonders whether the idea has ever occurred to their designers at all: the section header shall be physically closer to the next which it introduces than to the text which precedes it. Note that this is strictly a 'less than' relationship, not a 'less than or equals' one: the header must *never* be quasi-spaced between the preceding and following texts. This rule has some interesting knock-on effects: for example, a header must *never* appear in isolation at the bottom of a page, for were it so to do, it would by definition be nearer to the preceding text than from the text which follows.

[6] A concept whereby a ragged-right setting is used in conjunction with 'strongly recommended' line breaks, thereby ensuring that complete ideas (phrases, clauses, sentences, etc.) are not unnecessarily split over two lines.

But in a stronlgy hierarchical book or paper, it is just as important that the different levels of header (A-head, B-head, etc.) shall be capable of being differentiated at a glance. How should this hierarchy of headers best be conveyed to the reader? We have available several options: (1) Higher-level headers may be separated from their preceding text by greater amounts of vertical white space than lower-level; (2) Higher-level headers may be separated from their qualifying (following) text by greater amount of vertical white space than lower order; (3) Higher-level headers may be set in a larger font than lower-level; (4) Higher-level headers may be set in a bolder font than lower-level; (5) Some other typographic differentiation (e.g. the use of a *sans serif* font in a book or document othersie set of in a *serif* font) may be used for one or more levels of header; (6) Run-in headers maybe used for the lowest level of header. Indeed, these are only some of the available options: for example, in some works a new page is taken for each new top-level section, even where that section is only one of many similar sections in a chapter.

Clearly the range of options is vast, and it is not possible in a paper of this brevity to give more than a few typical conventions, but one requirement is tantamount: if two or more conventions are adopted within a single document, then no combination of those conventions must lead to ambiguity. For example, if A-heads are set in 16 point roman, B-heads mst not be set in 14 point bold, for the boldness of the B-head would counteract the effect the smaller font and lead to ambiguity in the mind of the reader. Even if a bold font is not explicitly used, it is possible (by, ofr example, selecting an ill-matched *sans serif* font for B-level headers in an otherwise *serif* document) to accidentally specify an *apparently* bolder font for a subsidiary-level header. Such ambiguities must be avoided.

In what other ways can the book designer ensure that information is most clearly conveyed? Perhaps most important of all by ensuring that the book can be *read*! This goes without saying, you may say, but there are sadly only too mnay counter-examples already published for this particular requirement to be omitted from any reasonably critical analysis. Perhaps we need to start by defining what we mean by "to read"; I suggest that if reading is to be conducted efficiently and pleasurably, then it must (for the normally-abled adult) be an almost unconscious activity. If I pick up a book hoping to gain information from it, then the *last* thing that I want is to have the designer's personality forced down my throat (unless it is a book on book design, in which case I may be able to judge from the book's design whether or not to bother to read it!); the design must therefore be very 'quiet' and unintrusive, allowing the content to flow naturally forth through the medium of the form, rather than having the form leap out from the page and distract the reader from the content. Naturally there are exceptions to this rule, and books on design clearly come into that category, being inherently self-referential, but generally speaking the reader wants to know as little about the designer and as much about the content as possible.

Furthermore, reading must be able to proceed in a linear and uninterrupted manner, it is well known that any infelicity on the part of the author which results in ambiguity in the reader's mind will cause the latter to back-track through the work, hoping to gain further clues and thereby disambiguate the text on a second or subsequent reading. Classic authors on grammar (Fowler, Weseen, Partridge, Onions, Gowers, Quiller-Couch, Sweet) pay much attention to this. But there are many typographic pitfalls which can also cause a reader to have to backtrack, and it is as important for the designer to avoid these as it is for the author to avoid the grammatical infelicities.

For example, during the 1930s, there was a great vogue for *sans serif* faces: they were modern, *avant garde*, stylish modish – use whatever term you will. And particularly in North America, and to a lesser extent in Europe, such was the pressure to use these typefaces that their raison d' être – to provide a simple, minimalist, style for short sections of text which would not draw attention away from the main theme (freauently an accompanying graphic) – were forgotten, and they were advocated (and used) as *the* typefaces for every conceivable purpose. These purposes were not restricted to their classic use in headings, captions, posters, etc., but were instead extended to encompass even the running text of books; every page was set in *san serif* text, with little feeling for the comfort and convenience of the reader. The effect on the reader was all too predictable (with hindsight): readers found it difficult to concentrate on such books for any period of time, finding it tiring and even distressing; and the reason was very simple, although not well understood at the time: even though the *serif*s which characterise most of our classic

typefaces today are in reality no more than artifacts dating back to the original letterforms of stonecutters (particularly) in the case of upper-case letterforms), and later of typecutters, these *serifs* perform a very important function when the letterform occurs in running text: they serve to draw the eye naturally along the line of text, very much reducing the risk of the eye vacillating between two adjacent lines of text, and also help to minimise the amount of backtracking within a single line. And so, with the benefit of hindsight and of psychological and physiological research, it has now been established that the typeface of choice for passages of running text (as opposed to captions, etc, which extend for at most a few lines) is almost invariably a *serif* face. Sadly this fact is still occasionally ignored.

But if the choice of a *serif* face is almost mandatory to avoid vacillation between lines of text and backtracking within a single line, what other psychological or physiological factors can also affect the readability of the text? Perhaps the most important of all, and one for which plain TEX sadly gives most inappropriate guidance, is the size of font with respect to the *measure* (*i.e.*, the width) of the text. Plain TEX is predicated on the use of 10 point fonts on a measure of 6.5 inches (39 picas), which simply gives far too many characters per line. Psychologists have shewn that the optimal number of characters per line for normally sighted people lies in the range 40–70, and peaks somewhere near the upper bound of that range; below it, people become frustrated: they are forced to take in too little information per glance; and above it, they tend to lose their place, and either backtrack within the line, or on re-scanning to the start of the next line, lose their vertical place and re-scan to the start of the wrong line. Even LATEX, which generally gives better guidance than plain TEX in matters of typographic design, allows the user complete freedom to select between 10 point, 11 point and 12 point fonts, regardless of the style chosen and therefore of the measure of the text. For Europeans readers, accustomed to the DIN series of paper sizes, the best guidance I can give is as follows: if you are setting on a sheet of A4 paper (which is unlikely for a book but quite possible for a report or other similar document), which 'normal' margins (circa 1 inch), then a 12 point font is called for; you can get away with 11 point, but 10 point is out of the question. The same goes for North American readers with 1 inch margins on a sheet of American 'letter' paper, $8.5'' \times 11''$. And for a book? Well, 'how big is a book' is a question to which I will return in the sequel to this paper, but generally speaking books *are* set in 10 points typefaces; however, as the width of the paper increases, two columns become obligatory or pathologically large margins become reguired.[7] In unusually small books, 9 point fonts may be used, but anything less than this poes problems of legibility for normally sighted people.

In the preceding paragraph, I have spoken of a "10 point font" as if it were some sort of ISO standard; but sadly it is anything but. Fonts vary enormously both in their actual size (as measured), and in their perceived size, and the quoted size is at best an approximation and at worst a d@mned lie! For what it is worth, the notional size of a font is that distance which may separate consecutive lines of text in a paragraph set in that font without the descenders of one line overlapping the ascenders of the line below; it is also approximately the height + depth of a parenthesis glyph. But in practice one designer's 10 point font may well be another's 11 point; and if you are using two or more fonts in a single document, then it is your responsibility as designer to ensure that the size at which they are used renders them visually conformabale, even if this means loading one at 10 point and another at 11 point (or even at 10.6347 point, if that represents the true ratio between their perceived sizes).

And for the leading: some authorities will suggest "1.2 times the design size of the font"; others will suggest "2 points more than the design size of the font"; and others will suggest yet further formulae. The answer is, of course, that no one formula will be right for every font, or for every size, and until experience has given you the insight to look at a font sample and *know* the appropriate leading for the target font size, then you will have to use the most powerful too available to you: your eyes. In other words, you will have to print samples of the text at various leadings (probably of the order of magnitude suggested by the formulae above), and adjust until it looks right to you. But when you print these samples, you will come up against another, and very subtle, psychological quirk: assume you do as most people

[7] I am advised by a North American student that it is the practice in North America for students to annotate their books; for this reason, they *expect* far wider margins than European readers, which may explain something about the default LATEXstyles.

do, and print your proofs on a laser printer; then your output will appear either on a sheet of A4, or on a sheet of 'letter' size paper, and most unusually on anything else. And try as you might, you will not be able to judge the size of the font and the size of the leading as they will appear in the final book form, even if you draw a box around your sample text to represent the dimensions of the final trimmed page; your eye/mind will refuse to believe that the white paper which lies outside that lines is not attached to the text, and will judge the size of the text and the size of the leading in terms of the untrimmed sheet of A4 or 'letter' paper. The solution, of course, is to guillotine the paper to the final trimmed size, and then to paste two such trimmed sheets together (or to print a double page spread in the first place) and to look at a full-size replica of the final spread of the book; and then, and only then, will you be able properly to judge the size of type and the size of the leading in terms of the printed page.

5 Structure

Finally we turn our attention to *structure*, and in particular to the means by which a well-designed book can be efficiently referenced (and cross-referenced) in a quasi-random, rather than sequential, manner. At the coarsest level of granularity, a book is divided into volumes (if huge), parts (if large) and chapters (almost all books). Access to volumes need not worry us unduly: each will contain the name and/or number of the volume on the spine and front cover, and only if two or more volumes are concurrently open in front of the reader will it be necessary to be able to differentiate between volumes by inspection of only the open spreads.

Parts are not uncommon, but many of the potential problems associated wth the identification of parts can be eliminated by sequential numbering of chapters independent of the part in which they happen to fall; with sequential chapter numbering, the reader can always be referred to *Chapter n*, without needing to quality it as *Chapter n of Part m*.

But the most important division of the majority of books is into chapters, and here we must start our investigations into *structure* in earnest. Consider the classic case of a multi-chapter, single-volume, book, with a table of contents ('TOC') among the *front matter* (a.k.a. 'the prelims'). The reader wishing to access the book through the TOC consults the latter and sees, for each chapter, its number, its name (if the chapters are named), and the page on which it commences. Selecting a chapter from those listed, the reader flicks through the pages looking for the page on which the chapter starts. This is not a random search: the page numbers increase monotonically with period I, and if the reader overshoots he or she is invariably sufficiently familiar with the general concept of a book to realise that it is necessary to backtrack.[8] But an interesting phenomenon occurs as the reader converges on the page of interest, at least in many less-than-optimal books: the page numbers (*folios*, as they are frequently termed) traditionally alternative between top-left and top-right, occupying the top-left placement on verso pages and top-right on recto; this placement is believed to make them maximally visible. But on opening chapter pages it is traditional to suppress the running head ('headline'), because the design of these pages (discussed in greater detail in the sequel to this paper) is such that a running head is generally considered aesthetically displeasing. And therefore the very page which (logically) bears the number sought is also the very page which (physically) has no page number on it; and the reader is forced to perform a narrow binary search to ensure that the page of interest has truly been located, by comparing the last physical page number which can be found (and which will, in the worst case, not even be visible from the page of interest, if the previous chapter happens to finish recto, since it is also traditional to start new chapters recto and a completely blank page will therefore form the vero half of the spread) and the next physical page number,

[8] It is interesting to realise that the scenario outlined is the converse of what usually happens in practice: because books are generally either laid on the desk/lectern or held in the right hand with the highest number page at the bottom, it is far more natural for the reader to make a *backwards* search through the pages until the desired page is found, or until overshoot coccurs, than it is to make a *forwards* search. This is because it is far easier to raise a number of pages, frequently almost the entire page set, in one hand and allow them to fall back individually under the effect of gravity than it is to lift each page individually whilst seeking the page of interest.

which will invariably also be invisible from the page of interest. Of course, the name and/or number of the chapter will be visible on the sought page, and it will be clear from its design that it *is* an opening chapter page, but none the less the reader who until then has been searching for a specific page number is forced to modify his/her search algorithm.

The solution generally advocated for this problem is to present the page number on opening chapter pages as a *drop (ped) folio*: a centered page number occupying a part of the footline. The percipient reader soon becomes familiar with this convention, and modifies his/her gaze to take in the bottom of the page rather than the top outside edge when reching an opening chapter page. But if dropped folios are acceptable on opening chapter pages, why not use them consistently throughout the book? This would have two beneficial effects: (1) the reader would be able to find *any* page in the book by studying the same part of every page, regardless of the nature of that page, and (2) additional space would be released in the running heads for additional (cross)-referencing material, space which as we shall see becomes of a premium as the complexity (in terms of explicit hierarchical structure) increases.

Once we have ensured that page numbers occur on *every* page (blank pages excepted, since by definition no possible well-formed (cross)-reference could require the reader to turn to such a page), we have at a stroke ensured that our tables of contents, indexes, etc., all of which generally yield a *page number* when 'dereferenced' (consulted), will invariably result in a hit rather than a miss. We must now turn our attention to other techniques for (cross)-referencing, and in particular methods for locating logical sub-divisions of the book (e.g. sections, sub-sections, etc.) by their *name*, and also by their *number* if such entities are numbered.

Generally speaking, the names and numbers of logical sub-divisions are used for cross-referencing (*i.e.*, referencing from within the text), rather than for direct referencing (e.g. from a table of contents or an index); but regardless of the source of the reference, the reader will ultimately be required either to *see Section 2.1.4* or to *see also Lagopus hyperboreus* – in neither case will be reader explicitly be instructed to turn a specific page. It is frequently possible to convert one of these *indirect* references into a *direct* reference to a page number, by consulting the appropriate table of contents or index, but this two-stage process is both frustrating and time-wasting: a more direct method is required.

The mechanism by which this direct access to named or numbered logical sub-divisions of a text is generally accomplished is through the medium of *running heads*; these have been referred to previously in the current paper without any formal definition being given of their nature or purpose. A running head is so called because it recurs on (almost) every page; opening chapter pages and blank pages are usually excluded from the set of pages and blank pages are usually excluded from the set of pages on which a running head can occur, and if an entire page is given over to an illustration then that page too may be excluded; but special cases apart, running heads occur on every page. But of course the *content* of the running head varies from page to page: were it not so, there would be no purpose to the running head at all (which is also frequently the case when it is used to echo the title of the book on every page or every second page; the reader is normally aware of the title of the current work, although there are counter-examples, as when consulting many works at once; thus the echoing of the title is not necessarily evidence of poor desing). In general, the content of the running head is adjusted to reflect the content of the page over which it appears; thus, for example, if *Section 2.1.4: metalinguistic notions* commenced on page 23, the running head of page 23 would almost certainly reflect that fact. But in a hierarchically structured work, there are potential conflicts; consider a book with chapters, sections and sub-sections: which of these entities should the running head reflect? A convention frequently adopted is to ascribe different semantics to the verso and recto heads: the verso carries 'more significant' information (e.g. the name/number of the chpater), whilst the recto head carries 'less significant' information (e.g. the section name/number). Yet this is not enought: where should the sub-section information appear? Ultimately there is no solution to this problem: if the book is sufficiently complex (*i.e.*, possesses too deep a nesting), then no matter how complex an arrangement of headers is adopted there will be a level of nesting beyond which it is simply not possible to reflect lower-order entities in the header. The designer, then, must perform a trade-off, and decide which information is most beneficial to the reader. Omissions are possible

at either or both ends of the spectrum: it may be that knowledge of the name of the current chapter is less important than knowledge of the current section/sub-section/sub-sub-section/sub-sub-sub-section; or it may be that knowledge of the chapter is deemed more important than knowledge of the current sub-sub-etc. The designer and author must work together on this problem.

But there is one additional mechanism which is considerably under-used, yet which allows twice as much information to be packed into each header. If folios are removed to the footline, thereby releasing the outer edge of each running head for other usage, then provided that the author can be encouraged to provide *short* names for each of his/her chapters/sections/etc., each running head can serve double duty. For example, verso heads can carry (left) chapter name/number, whilst carrying (right) section name/number; recto heads can then carry (left) sub-section and (right) sub-sub-section. Adequate space must clearly be left between the two elements to avoid potential ambiguity.

Finally, is it the *name* or the *number* of each logical entity which is to appear in the heder? Above I have hedged my bets by consistently referring to name/number, yet at some point a decision must be made. If space allows, and if the author co-operates by providing short names, then there is no reason why *both* should not appear; with less space, or longer names, it may be necessary to omit the numbers in order to allow the names to appear; and if the author is unconscionably prolix in naming the various entities, then the designer may have little choice but to simply give the hierarchical name (e.g. *Chapter*, *Section*) followed by the relevant number. But this last serves the author rather than the reader, and pressure should be brought to bear on the author to provide suitable ‘short forms’ purely for use in the running heads. Of course, some works use *only* numbered entities; in such works, there is no choice: the hierarchical names (if appropriate) and numbers must be used.

6 Conclusions

Good book design can be discussed in terms of three parameters: *uniformity*, *information* and *structure* (although there are many other parameters which would be addressed in a longer paper), and attention to each of these will do much to increase the potential value of a book to its readers. More practical advice is given in the sequel to this paper: “Book Design for TEX Users; Part 2: Practice”, elsewhere in this volume.

TUGIndia Journal **1** (1): 36–39 (1998)

The inaugural meeting of TUG India

Sebastian Rahtz
Elsevier Science Ltd, Oxford, UK
s.rahtz@elsevier.co.uk

1 First stirrings

Back in the summer when I first started corresponding with C. V. Radhakrishnan in India about TEX and SGML-related matters, I little thought that I would be escaping the English winter for a week in Southern India at the start of 1998. But something seemed to crystallize in the minds of some Indian TEXies, and events moved fast in the subcontinent during the autumn. By November 16th, Radhakrishnan was able to announce to the world that the newest TEX user group had been born:

> The Indian TEX Users Group has been informally launched today at the academic premises of Department of Mathematics of University of Kerala, Trivandrum. Prof. KSS. Nambooripad, a world renowned mathematician and an ace TEX programmer chaired the session. He was unanimously elected as the Chairman of the Indian TEX Users Group. Following are the office-bearers of the TUGIndia.
>
> | | |
> |---|---|
> | Chairman: | Prof. (Dr.) K. S. S. Nambooripad |
> | Secretary: | C. V. Radhakrishnan |
> | Treasurer: | Dr. R. Rajendran |
> | | |
> | Executive: | Dr. A. R. Rajan (University of Kerala) |
> | | Dr. E. Krishnan (University College, Trivandrum) |
> | | Dr. V. N. Krishnachandran (Vikram Sarabhai Space Center) |
> | | Dr. R. K. Chettiar (Department of Education, Govt. of Kerala) |
> | | Mr. C. V. Rajagopal (University Observatory) |
> | | Mr. Deepak Tony Thomas, Oracle Corporation, Bangalore |
> | | Dr. P. Rameshkumar (MG University, Kottayam) |
> | | Dr. SRP. Nayar (Inter Univ. Center for Astronomy, Pune) |

At the same time, they did me the great honour of inviting me to inaugurate the group, and I lost no time in accepting in principle. In the ordinary course of events it would have been beyond the finances of either TUG India or myself to pay for a trip there, but then a fairy godmother appeared, in the shape of the UK TEX Users Group. The committee considered my tentative suggestion, and agreed that support of such a potentially important group would be a reasonable use of group funds. That just meant fixing a date, and finding a flight, and all was in train. By an amazing coincidence, another member of the

UK TUG committee, Kaveh Bazargan, had already booked a holiday over Christmas and New Year at precisely the location in India chosen for the TUG India launch, so we were able to mount an even more impressive presence.

2 India, and the inauguration

The TUG India meeting took place in Trivandrum, the capital of Kerala state, which forms the southwest corner of India. It is a tropical area, which sometimes seems to be entirely covered in coconut palms and banana trees, and is famous for its communist state government, its almost 100% literacy rate, and a general air of some prosperity and a good distribution of wealth.

The principal mistake I made before setting off was to contract a vile cold, which rendered me almost speechless during my first few days, and a poor picture of health for the whole stay. However, after a long flight from London, a sweaty wait in Mumbai, and then a short flight to Trivandrum, it was little hardship to be taken off after lunch to the excellent beach resort of Varkala. Talking TEX beneath the palms next to a sunny beach of the Arabian sea was a little disconcerting, but we managed...

On Monday January 5th, we adjoined to the University Observatory (a purely courtesy title these days) for the opening ceremony of TUG India, at which Kaveh and I were joined as speakers by Professor Nambooribad, the group's chairman, the University Vice Chancellor, and the local member of Parliament (showing a healthy interest in IT matters). Kaveh and I tried to present the TEX world as place of dynamism, excitement and new possibilities for conventional and electronic publishing, and some at least of our audience seemed convinced.

Many of the delegates were from typesetting companies (some of them suppliers to my own employer, Elsevier), with the biggest contingent from Madras — Thomson's office seemed to have sent almost all its R&D team. But there was plenty of academic interest too, and of course a special concern with typesetting Indian scripts.

It was a pleasure to be able to hand over to Radhakrishnan a selection of TEX-related books and journals, donated by Jonathan Fine, Malcolm Clark and myself, and to confirm the imminent despatch of back issues of TUGboat and MAPS to India. NTG had already sent a generous batch of 4AllTEX CD-ROMs, of which each delegate was given a copy.

3 The first TUG India courses

On January 6th the serious work started, four days of tutorials in the morning on 'advanced' topics, and introductory LATEX in the afternoons. I managed to avoid teaching the latter (I always find myself *apologizing* too much for LATEX), but had fun in the mornings.

We started by dealing with a subject dear to my heart, and to that of some of the delegates — LATEX to SGML translation. I expounded the Elsevier system, based on four stages of transformation:

1. LATEX to dvi, using a very specialized class file, which redefines almost everything to put SGML markup into the dvi file;
2. dvi to ASCII (using Tobin's dtl programs);
3. ASCII to SGML against an intermediate DTD;
4. SGML to SGML for the final DTD (using a Perl library with directly interfaces with the NSGMLS parser).

It turned out that at least two others present had also thought of similar methodologies, which was reassuring.

From LATEX to SGML, I moved on next day to DSSSL (Document Style Semantics and Specification Language) and its relationship with TEX — perhaps not everyone present quite went along with me on that one. We were on safer ground discussing general aspects of electronic publishing using TEX, and I

was glad to able to be describe pdfTEX in some detail, to publicize what I consider a much under-rated alternative to LATEX2html (Eitan Gurari's TEX4ht), as well as give a puff for my own LATEX hyperref package.

On the third day, we moved onto pictures, and I attempted to make a (rather shaky) case for MetaPost. Colour was a subject where it was easier to find common ground, albeit by agreeing that color separation specification in TEX was much too immature at present.

For the last day, I had decided that this was the moment where I would really make a first go at using Omega, and (somewhat to my surprise), I was able to write, compile and use a one-line Omega Transformation Process after some study of Omega examples. Since one of the Omega authors (Yannis Haralambous) is very actively working on the necessary OTPs, hyphenation and so on for typesetting Malayalam (the language spoken in Kerala), we can expect rapid deployment of Omega amongst those typesettings things like school textbooks.

4 . . . and some sightseeing

After talking TEX for 5 days, I was ready for some relaxation. We started with a shopping expedition, during which I bought some dresses for my daughters which are certain to lighten up wintry Oxford, and a selection of South Indian classical music. Then on the Saturday we drove across the state line into Tamil Nadu to visit the Padmanabhapuram palace of the Maharajah of Travancore, the princely state which occupied much of what is now Kerala until Independence. In the late 18th century, a replacement palace was constructed in Trivandrum, and Padmanabhapuram was left untouched. With elements from the 16th century, it is an incredible structure built almost entirely of teak, often intricately carved, and all ingeniously designed to keep the rooms cool with natural air-conditioning. Whether it was the ladies bathing tank, the audience chamber, or the hall where 2000 Brahmins could be entertained to dinner, the whole place was a marvel of design — and preservation by the State Archaeological Service! Perhaps the best moment was when we were granted special access to view the Maharajah's private meditation apartment whose plaster walls were covered in marvellous paintings, and where a pair of coconut-oil lamps had been burning non-stop for 200 years.

From the past to the present, as we drove to Cape Comorin, the southern-most tip of India, where you can see both sunrise and sunset across the sea from the same spot, and where three oceans meet. Here, in the late 19th century, a man who became a very influential religious reformer swam out to a bare rock in the sea, meditated for five days, and achieved a state of enlightenment to accord him the status of a saint. Now there is a modern memorial on the rock, and we joined hundreds of pilgrims in the boat ride to examine the spot. Thence back north, trying to visit a Jain temple set deep in a cave, but sadly the gates were locked, and some monkeys laughed from the rock.

On the Sunday, to Kerala's secret paradise, the long salt waterway that runs for 200 km parallel to the sea, sometimes as a wide as a lake, at other times turning into quiet green tunnels with barely enough depth for the boat. A vista of endless coconut palms, half-hidden houses, and small fishing boats provided a very relaxing boat trip.

5 Conclusion

This was a worthwhile, if exhausting, trip, and I hope it gave a good start to TUG India. When I left, they already had 79 members signed up, just from word of mouth, so the group looks set to be active. It is hoped to cycle the meetings around the different parts of India, as well as publishing a newsletter, so the current bias towards the south should soon be corrected.

I must, of course, take this space to extend the heartfelt thanks of Kaveh and myself to Radhakrishnan and the many others who looked after us so magnificently during our stay in Kerala. They were very

worthy ambassadors of a lovely part of India. I look forward to working with them, and hopefully to visiting India again soon.

TUGIndia Journal **1** (1): 40–43 (1998)

A Case for TEX in India — The Indian TEX Users Group

C. V. Radhakrishnan
River Valley Technologies, Software Technology Park
Trivandrum 695034, India
e-mail: river@earthling.net

Abstract: Unlike other countries, TEX migrated to India as a medium of typesetting for the Western publishing world. With its vast human potential and cheapness of its cost, India enticed the publishing giants like Elsevier, Academic Press, Springer-Verlag, etc., for their pre-press work and with that TEX language found its way into this subcontinent. Its meager presence in the very many higher academic institutions and its pronounced absence from ordinary institutions strengthens the paradigm that the TEX usage in India is primarily a gut-oriented phenomenon rather than an author driven one. This is further demonstrated by the clear absence of TEX related research, newer macro development, font generation for the multitude of Indian scripts, etc. The Indian academy, contrary to its Western counterparts, pays scant regard for such things or seldom considers it as a necessity. Therefore, the problems of TEX usage in India is diametrically different from that of other parts of the world. It is not strange that the Indian TEX Users Group, formed recently, faces the constraints of lack of research and economic issues of the users as well (quite strange!) since the vast number of its members are from the typesetting industry who chose TEX language as a means of their livelihood. These and related issues are described in this article.

1 A general overview of TEX users in India

There appears to be a vertical split when we consider the general users of TEX language in India, one from the typesetting industry and the other from the higher institutions of learning. The former category may outnumber the latter. Except for Indian Institutes of Technology (a chain of institutes spread all over India, noted for its academic excellence and standards) and certain specified scientific institutions like Indian Institute of Science, Inter University Center for Astronomy and Astrophysics, Tata Institute of Fundamental Research, etc., TEX is still alien to the academy or the researchers. The advent of WYSIWYG typesetting software has further pushed back the chances of TEX usage. Yet another paradoxical element you can observe here is that the Indian academy considers typesetting issues as the burden of publishing houses and it is not the concern of the author, to address the various enigma of his own document presentation. This is the general philosophy of even the computer scientists working in various Universities in India. The limits of our document preparation skills are dictated by few Microsoft products. If any of the Microsoft product is incapable of presenting our document, we would resort to manual operations thereby making it a childish doodle, for, the present day Indian mind is not at all carried away by refined and sophisticated presentation, in sharp contrast to the classic Indian aesthetic sensibilities.

This article is reprinted with the permission of TUGBoat wherein it first appeared in the March 1998 issue

This being the general attitude of the academics around this country, the quantum and the quality of TEX usage in the academy can be gauged by anybody. This may be the prime reason why India lagged behind in forming its user group when all the rest in the educated West went ahead with their user groups and made substantive contributions to TEX language. India became a silent spectator, with a subdued longing for enjoying the fruits of TEX research in the West with an apparent resignation that is typical of a Hindu mind. The shape of things in the Indian typesetting industry is also not so bright. Due to lack of any meaningful research and development team, they solely depend on or unabashedly hire Western intelligence for the development of their in-house arsenal. Even in matters as simple as writing a filter for SGML to TEX or *vice versa*, they do get filters written by external agencies, present to their clients as if developed by their own R&D team and win huge contracts. In short, healthy usage of TEX language is still a distant dream in any of these agencies.

2 Early work on Indian languages

One of the earlier work on TEX language that concerns Indian scripts are done by Avinash Chopde[1] and the package is called ITRANS which bundles lot of Indian scripts with LATEX. You create a `.itx` file and run thru ITRANS to convert it to a `.tex` file. The commands are same for Tamil, Sanskrit, Marathi, etc. His home page describes the system fully, and is available for UNIX and PC platforms.

There is also JTRANS (Sandeep Sibal) a Java program that enables you to see Sanskrit text in an html document. There is also an `Xdvng` font that if installed will view Sanskrit documents on the web. All these are explained in detail in the file `index.html` available via `anonymous ftp` from `jaguar.cs.utah.edu` in the directory `private/sanskrit`. You will have fun with all these programs and Avinash can throw more light on all these topics, since he created the various ITRANS versions. He has also an ITRANS songbook that lists several thousand Hindi film songs in Devanagari script.

If you visit `http://www.concentric.com/ Dchand/jaguar` and click on Processing Tools, where several packages for processing Sanskrit on the net are described with pointers to ITRANS, JTRANS. Currently ITRANS supports Devanagari (Sanskrit/Hindi/Marathi), Tamil, Telugu, Kannada, Bengali, Gujarati, and Romanized Sanskrit script output.

The input text to ITRANS is in a transliterated form, each letter in an Indian Script is assigned an English equivalent, and the English letters are used to construct what will eventually print out in the Indian Language Script.

ITRANS offers a choice of two input encodings: ITRANS encoding, and the CS/CSX encoding. ITRANS encoding is a 7-bit ASCII encoding, while the CS/CSX encoding is a 8-bit encoding. The ITRANS encoding requires multi-character English code be used to represent each Indic Script letter, while the CS/CSX encoding uses a one-character code to represent each Indic script letter.

Apart from this other meaningful work undertaken in TEX related area are various fonts created using METAFONT or tools like that. Some of the work has been undertaken by non-Indians too.

1. ItxGuj, a Gujarati font, and ItxBeng, a Bengali font added to ITRANS. These fonts have been donated to ITRANS by Shrikrishna Patil, and are available in PostScript Type 1 and TrueType formats, so can be used for printing as well as for display on WWW browsers such as Netscape 3.0 (or later).
2. Though lot of improvisation is needed, KannadaTEX developed by the Central Institute of Indian Languages, Maysore is a commendable work in the right direction. For Kannada font from the KannadaTEX package, ITRANS support was added by Raghunath K. Rao. This is a font in METAFONT format, so can be used with TEX only.
3. Devanagari font: `Xdvng`, by Sandeep Sibal. available in PostScript Type 1 and TrueType formats, so can be used for printing as well as for display on WWW HTML browsers such as Netscape 3.0

[1] `http://www.paronia.com/~avinash/itrans.html`

(or later). The `Xdvng` font is a derivative of the `Devnag` font that has been developed by Frans Velthuis.

4. Romanized Devanagari fonts: `CSUtopia`, by Dominik Wujastyk,[2] and Washington Indic Roman by Thomas Ridgeway; both in Classical Sanskrit Roman encoding (CS/CSX encoding).
5. Malayalam font: by Jeroen Hellingman[3] which is a commendable work for both the traditional and reformed scripts. This is complete except for METAFONT sources; instead a range of pre-compiled sizes is included for the main font is available at CTAN. This system comes with two pre-processors `patc` and `mm`.
   The malyalam.sty[4] package is an interface to malayalamTEX, for use with LATEX 2ε. It works by loading Hellingman's macro files `mmmacs.tex` and `mmtrmacs.tex` to interpret the TEX macros generated by the `patc` and `mm` pre-processors.
   Note that these macro files are *not* provided as part of malyalam.sty, but must be collected separately from CTAN or elsewhere.

3 Problems of TEX in India

As you can see, except for a baseline research on some font generation, nothing substantive is forthcoming from the Indian TEX world. TEX have never percolated into the local publishing industry. As such, document preparation, (especially technical documents) in the regional languages suffers considerably and its current status is deplorably poor. It has an indirect effect on the development of scientific document presentation in local languages. People quite simply are forced to believe that our languages are not fit for scientific document presentation and console themselves that it is a cherished domain of European languages. For instance, the State Languages Institute of Kerala (Kerala is one of the Indian States where literacy rate has surpassed 95%) which is the official body for the production of school and University text books in Malayalam (the language of Kerala) finds it difficult to produce advanced scientific books with a quality comparable to English language text books, though intellectual resources are abundant.

Secondly, with a very healthy and vibrant literature, the Indian regional languages publishing is one of the richest industries in the country. But electronic digitizing and archiving of multitude of books released in a variety of languages (both officially recognized and otherwise) is a distant dream for us. No effort has been invested to address the problems of archiving of text data and its retrieval. SGML (Standard Generalized Markup Language) is still alien to Indian languages. A vast heritage of Indian Literature still thrives on paper which raises multitude of issues relating to storage and retrieval of information. The advent of INTERNET and WWW has prompted very many Indian regional periodicals to enter into the world of electronic magazines. Without proper fonts and encoding scheme fit for the WWW browsers, most of them are still wallowing in the primitive world of presenting images of whole text pages which becomes highly unpopular among the Indian viewers where the poor dialup line speed prevents easy browsing.

Thirdly, the wide gap between the Indian TEX Users and the current status of TEX in the world. The old LATEX 2.09 is still in popular usage among most of the typesetting houses and general users as well. Most of the users are afraid of LATEX 2ε. When the world is hopefully anticipating the arrival of LATEX3, our users are still in the domain of the obsolete LATEX 2.09. In the workshop held along with the inauguration of Indian TEX Users Group, most of the participants had not heard of graphics inclusion programs like METAPOST, XYPic, PSTricks, etc. PostScript and its relationship with TEX is also at a distance.

[2] `d.wujastyk@ucl.ac.uk`
[3] `etmjehe@etm.ericsson.se`
[4] `http://tug2.cs.umb.edu/ctan/tex-archive/language/malayalam/`

4 Formation and relevance of Indian TEX Users Group

It is at this state of affairs, some of the TEXies in the southernmost part of India came up with the idea of forming a Users Group in India. Sebastian Rahtz[5] of UKTUG played a key role in its formation. Few of the academics and researchers from the University of Kerala, Trivandrum, scientists from the Space Center, programmers from Indian software and typesetting companies assembled together and launched the Indian TEX Users Group (short-named TUG*India*). The aims and objectives of TUG*India* do not differ much from that of the international TUG with special emphasis on extending TEX to Indian languages. As a first step towards this goal TUG*India* is associating with Yannis Haralambous[6] of French TEX Users Group to build Omega-Malayalam system. Preliminary work done so far gives enough and more encouraging results and with these results the local education department has accepted in principle to finance projects relating to the actualizing and perfecting Omega Malayalam system that can solve the problem of technical document preparation in Malayalam.

This is only a first step towards extending TEX to Indian languages. Slowly and steadily this mission would be spread to other parts of India to cover all the major languages. It may sound little strange whether a single user group in a vast country with diverse lingua and cultures can hold all the users with different identities together. Unlike other parts of the world, this is an amazing truth so far as India is concerned and the TUG*India* Borad has decided that its secretariat would be shifting its location to different centers in India in a fixed periodical manner so that the current bias towards the south will be annulled.

To familiarize with the emerging trends in TEX research, TUG*India* would be holding periodical seminars and workshops, etc., and the first of its kind was conducted along with the inaugural ceremony. The main themes discussed were LATEX to SGML conversion strategies, pdfTEX and related issues, Hypertext in TEX, graphics and color inclusion in LATEX and MetaPost and other graphics programs.

5 Miscellany

Majority of TEX users in India is from the typesetting companies and the TEX implementations are naturally the choice of their employers. Most of them are using Y&Y with Win95 operating platform except for one company (to my knowledge) which uses *Textures* for Mac. Still another company which has more than 100 terminals employ Novell Netware and DOS based TEX implementation too. Most of the computers used are Intel based PCs. Unlike this scenario, the academic institutions like Indian Institute of Technology, Indian Institute of Science, Inter-Univ. Center for Astronomy and Astrophysics, etc., where TEX remains a leading document preparation medium, the operating platform is various flavors of UNIX. Most of the institutions are having Sun workstations, DEC Alpha systems, HP workstations and TEX implementations for these systems are in use.

6 Epilogue

The Indian TEX Users Group would be publishing a journal *viz.*, TUG*India* Journal every four months and the first issue is getting ready to be released within a fortnight. TUG*India* welcomes articles from all the TEXies interested in publishing his/her article in our journal. Various User Groups are also informed that TUG*India* is only happy to reprint each other's articles on a reciprocal basis. The address for communication: Indian TEX Users Group, TC 24/548, KRIPA, Sastha Gardens, Thycaud, Trivandrum 695014, India. Tel. +91 471 324341. Fax. +91 471 333186, e-mail: `tugindia@mailexcite.com`.

[5] `s.rahtz@elsevier.co.uk`
[6] `yannis@pobox.com`

TUGIndia Journal **1** (1): 44–46 (1998)

The Inaugural Address by Sebastian Rahtz

Vice-chancellor, ladies and gentlemen. Welcome to TUG India, the inaugural meeting of the Indian TEX Users Group.

It is a special honour for me to be here today because I am able to represent the UK TEX Users Group, formed nearly ten years ago, and foster the special relationship between India and the United Kingdom. Perhaps not everyone here realizes the extent of the reverse colonisation which India has had in England, influencing our culture, our food, our language and our attitudes. To me at least, coming to India is like revisiting ancestral memories. On a more personal level, I also recall my first visit to India, when I was visiting institutes and fund-giving bodies as a messenger of the World Archaeological Congress, talking about IT in archaeology. At that time, in 1991, I stopped over in Trivandrum *en route* for Sri Lanka, and passed one of the most peaceful nights I can recall in a pleasant hotel at Kovalam Beach. Since then, it has been my ambition to return to such a friendly place, and what better reason to come than to inaugurate the Indian TEX Users Group? I am very grateful to Professor Nambooripad and Radkrishnan for inviting me, and to the UK TEX Users Group for paying for my travel here.

Let us now turn to TEX. Perhaps it would be appropriate to give some history, since not everyone will be aware of the full story.

TEX was developed in the late 1970s by Professor Donald Knuth at Stanford University. He started what he thought would be quite a simple task of writing a typesetting program which he could use to prepare volumes of his monumental series, the *Art of Computer Programming*; in the event, it called for 10 years work, and the creation of not only TEX, but a companion font-drawing program, METAFONT, and the family of Computer Modern fonts. During the years of development, Knuth made some significant decisions:

- He decided to place the program in the public domain for all to use; not only was this generous in itself, but it prompted many people to try it, confident in the knowledge that they could find out what it was doing, that it had no secrets.
- He worked on the design substantially alone, albeit with a group of graduate students; this meant that the main TEX programs have a single architecture, and are to all intents and purposes free of bugs, the work of an individual not a team.
- Knuth listened to feedback, and encouraged it. This led to the creation of TEX Users Group in about 1980, which did much to popularize the software.
- He wrote the programs using his Web system of literate programming, in which the program code and its documentation are inextricably woven. The closely-documented nature of the code has made it possible for others to understand it. He even *published* the source code in a book, an almost unprecedented action.

This is the inaugural address delivered by Sebastian Rahtz on January 5, 1998 at the Observatory Seminar Hall, University of Kerala, Observatory Hills, Trivandrum, India.

- He did not commit himself to a specific output device (unlike troff, which did not achieve the same success as TEX), but separated out the 'driver' functionality into external programs, by defining TEX's output as a neutral device-independent format.
- He recognized that if the program were to have a long shelf-life, he could not commit himself to the graphical and color capabilities of the period. A conscious decision was made to *omit* any specific functionality of this kind, and instead to provide an all-purpose `\special` command.
- Lastly, Knuth froze TEX when he felt it was complete enough. This means that it stays forever as a fixed reference point, generating the same output from the same input and exhibiting no known errors. This allows some to build on it with confidence, others to write their documents knowing that they will always be able to processs them.

But things have moved on since Knuth wrote TEX; nowadays the talk is of desktop-publishing, word-processors, HTML, the Internet, PDF, SGML — is it an anachronism to start a new TEX Users Group in India in 1998? No, I believe it is not, and I have followed Don Knuth's favourite game of playing with numbers to explain why. Since this is (just) 1998, I have attempted to find 9.8 reasons why TEX is still relevant today, and why the Indian group will have much to contribute towards, I will go over my 9 reasons as reflections on the way TEX and its users are *evolving* to meet today's needs.

1 The Internet. The TEX user (well, LATEX user really, since it only makes sense if you use structured markup) can interact with the Internet in a variety of ways:
   - Converting LATEX into HTML, using latex2html, Tex4ht, or a variety of other programs.
   - Having LATEX read directly by a browser add-on, of which IBM's techxplorer seems to be the most fully developed.
   - Producing Portable Document Format for display using Acrobat Reader inside a Web browser. Using Han The Thanh's TEX variant, pdfTEX, or a dvi to PostScript program with Acrobat Distiller, we can make PDF files; even better, we can add macros like my LATEX hyperref package, or Hans Hagen's CONTEXt, and automatically produce PDF of a richness unsurpassed by almost any other program.
   - There is also now XML making a very welcome appearance in the Web world, and its math application, MathML. Can we hope for a happy co-existence between MathML and TEX?
2 Unicode, and its superset standard ISO 1646. 16-bit encoding is a reality, and is alive and well in the TEX world, with the extension of TEX called Omega, by John Plaice and Yannis Haralambous. Among many other things, this will allow Indian TEX users to typeset scripts like Malayalam to a high standard without recourse to pre-processors or any limitations on transcriptions.
3 Font control. Perhaps we are not in an ideal situation with regard to font encoding and usage in normal TEX, but we are getting there. While in the LATEX world T1 encoding is widely accepted, we can hope for similar agreement on 8-bit math font encoding.
4 Pictures. Has the TEX user ever had such a rich variety of ways to draw high-quality pictures? Whether it be MetaPost, XYpic, or PSTricks for general purpose drawing, MusixTEXfor music, PPCHTEX for chemisty, FeynMF for Feynman diagrams, or dozens of others,
5 Macro packages. With the decline of most of the original macro packages put together in the mid-late 1980s, LATEX, with its huge collection of add-on packages, now has a slightly unhealthy domination of the TEX-using world. Perhaps 1998 will be the time for some new competitors, like CONTEXt, to start a serious challenge?
6 TEX extensions. For a good many TEX users, all that is really needed in the Plain format, and a few (just a few) tweaks to TEX to solve some of those nagging programming issues. For some other users, what is needed is a complete reimplentation of TEX in Java, to provide the basis for serious experimentation with modularisation and spreading resources across the Internet. The first group

can rejoice, as the second version of ϵ-TEX is almost released, offering a variety of new features, without compromising TEX's integrity in any way. The second group can also start to uncross their fingers, as the NTS group will start work on a rewrite of TEX in Java during 1998.

7 TEX implementations. Does any TEX user still need to compromise on their implementation? With the standard Unix web2c implementation now available for DOS, Windows 32, Amiga, and any Unix platform, healthy competition in Windows 32 from MikTeX, ably maintained shareware Macintosh systems (OzTeX and CMaceX), it almost seems an embarassment of riches to mention the commercial Y&YTeX, Textures, and Scientific Workplace, each offering very significant advantages.

8 The relationship with SGML. We are reaching an understanding of how to use TEX as a backend formatter for SGML, and also how to use it as a front-end, by converting TEX inputs to SGML. Not an easy area to work in, perhaps, but TEX can hold its head high, and say that it is not afraid to work with SGML.

9 DSSSL, the ISO 10179 standard for a 'Document Style Semantics and Specification Language', is now not only a reality, but in daily use, thanks to James Clark's free implementation, jade. By separating style specification from paragraph makeup, this keeps a firm place for TEX in the universe, since DSSSL implementations (or the subsets, like XSL, which may be implemented for the Web) need a batch-oriented formatting engine to assist them.

9.8 User groups. Not a whole reason, because we do not yet have user groups to help TEX users all over the world. But by adding the Indian TEX Users Group today, we are making a significant contribution to filling that missing 20%.

In all of these areas, the point I want to stress is that TEX is standing up and doing taking its place in the modern world, while not straying from the firm basis of compatibility with the huge number of existing documents. TEX is not just a system that is good for typesetting academic maths articles, it is a general formatting tool ready to take its place in the next century.

I bring to the this event felicitations and congratulations from the Board of the TEX Users Group and the UK TEX Users Group, good will from the user groups of Holland and the Czech Republic, and the special 'good luck' from Barbara Beeton, doyen of TEX journal editors. I also bring with me in more tangible form gifts of books from Malcolm Clark, Jonathan Fine and myself from the UK, and as many back issues of various TEX journals as I could carry. I am delighted to see that the Dutch TEX people have contributed copies of the famous 4AllTEX CD-ROM.

To conclude, may I wish those assembled the best of luck in their task of spreading the word throughout India. Take TEX into new areas, develop it further, take a pride in it. But a word of warning to conclude — do not take TEX into a ghetto. Let it work *alongside* other tools and technologies, not against them.