Tutorials

Essential NFSS2, version 2

Sebastian Rahtz

Abstract

This article offers a user's view of the New Font Selection Scheme, version 2. It describes the reasons for using the NFSS; the differences a user will encounter between NFSS and old LaTeX; what it will be like installing and using NFSS2; some special situations in mathematics; and an overview of the advanced NFSS2 commands for describing new fonts.

1 Introduction

TeX has a very general model of using different typefaces; all it needs is to have a set of metric information for each font, and it leaves the actual setting up to the dvi driver. Within the TeX input file, the user associates a name with a given font at a particular size, and uses that name when she wishes to set in the font. In Knuth's descriptive 'plain' macros, this is easy to work with; in LaTeX, however, the user works at a higher level of commands like `\bf` or `\huge`, and the system is supposed to work out which particular font is meant. In the early 1980s when LaTeX was being developed, there was not much choice in typefaces (everyone used Computer Modern) or sizes, so Lamport simply 'hardwired' all the allowed combinations of size and style, and assigned them to specific external fonts.

Unfortunately, LaTeX's font selection is not as general as it sometimes seems. The commands do not add together to produce an intuitive effect; thus the commands `\it` and `\bf`, which separately produce italic and bold, do *not* produce bold italic when used together, but have the effect of whichever command comes last. Similarly, a size change always reverts to a normal weight font, so that `\bf\Large` does *not* produce large bold, whereas `\Large\bf` does. What is much worse is that if one wishes to use a different typeface throughout a document wherever (say) sans-serif is used, it requires major surgery to the innards of LaTeX; the font assignments are made in the file `lfonts.tex` which is read only when one is building the format file for LaTeX, something which most users never do.

The average user who simply uses the fonts which come with a typical TeX installation is not usually aware of the underlying lack of flexibility; however, now that many more fonts are available for use (a greater choice of METAFONT sources, and the ubiquitous PostScript fonts), this area has been a bottleneck in the use of LaTeX in general typesetting. The user who wanted Times Roman as the default typeface in a document had three choices:

1. To load many new fonts with the `newfont` command, and use them explicitly;
2. To heavily edit the file `lfonts.tex` and build a new version of LaTeX; or
3. To take large chunks out of `lfonts.tex` and replicate their effect in a style file.

Other style files were written to specifically implement features like bold sans-serif; but an altogether more general solution was needed, and this was met in 1989 with the New Font Selection Scheme, which was subsequently also used in $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX. The NFSS is a complete replacement for `lfonts.tex`; it contains a generic concept for varying font attributes individually and integrating new font families easily into LaTeX.

Since the first NFSS, considerable progress has been made towards a completely new version of LaTeX itself, and a new version of NFSS is one of the first useable results; this is what is described here.

2 Overview of NFSS2

The NFSS concept is based on five *attributes* of a font; the user changes any of the attributes, and it is the job of LaTeX to identify the appropriate external font which fits the new situation. The attributes are

Encoding The way the font is laid out; when only CM fonts are used, this does not vary, because all of Knuth's fonts have more or less the same layout. But users of PostScript fonts, or the recent 256-character 'DC' fonts (described in detail in 2), will meet different layouts. This is discussed in more detail in Appendix 1.

Family The basic design of the typeface, e.g. Times Roman, Computer Modern, or Stone Informal.

Shape The main divisions of the typeface; most have at least two variants, 'normal' and italic, and possibly something specialized like small caps.

Series Many typefaces come in a range of 'weights' (light, normal, bold, etc.), determined by the thickness of the strokes of letters; they also vary in width, as we see in Computer Modern, which has a normal bold and an 'extended' bold. The two attributes of width and weight are combined in the NFSS, and called 'series'.

Size The base height of the letter shapes (10pt, 14pt, etc.). These different sizes may be

achieved in two ways — by having a separate font design for each size, or by scaling of a single (usually) 10pt design. Typical Computer Modern and PostScript setups use a combination of methods, but we do need to worry about this for the NFSS.

At the start of a LaTeX job, each of the five attributes has a default value, typically as follows:

encoding	`OT1`	Normal TeX encoding
family	`cmr`	Computer Modern Roman
shape	`n`	Normal (upright roman)
series	`m`	Medium weight
size	`10pt`	

Inside LaTeX, this combination is looked up in a table to see which external font this situation corresponds to, and the result will be the familiar `cmr10`. Suppose we now wish to move into italics; we simply change the 'shape' attribute to 'italic', and LaTeX reconsults its table to find this means it should load and use the font `cmti10`. A subsequent request to change the 'series' to 'bold' will *not* change the italic *shape*, but search the table for a new combination of 'bold series' and 'italic shape', coming up with `cmbxti10`. Similarly, a request to change the font size at this point to 24pt will leave all other characteristics unchanged, and simply load `cmbxti10` at a larger size (since that particular font is available only in a 10pt design).

In practice, all this selection is hidden from the user behind familiar commands like `\em` and `\Large`, but it is important to realize that the scheme is completely general. If, for instance, we had available the full Univers family, which has a great range of weights, we could select any of them by changing the 'series' characteristic, and leaving LaTeX to find the right font metric file.

How does LaTeX know how to match a combination of `OT1+cmr+m+n+10pt` to `cmr10`? By means of special `fd` (*font description*) files (with a suffix of `.fd` in most operating systems). There is one of these for every combination of *font encoding* and *family*, which defines the possibilities of the other three attributes. Thus `OT1cmr.fd` classifies all the normal Computer Modern fonts by means of the font attributes, and says which external font they correspond to. When LaTeX starts, it does not have *any* fonts loaded[1], but waits until some font selection is needed, and then loads the appropriate `fd` file; as new combinations of font attributes occur, `fd` files are loaded. We will see later what the format of these font files is like.

[1] Not quite true; it needs some fallback defaults if all else fails.

Perhaps the most dramatic effect of attribute changing is when a different *family* is selected, and with a single command the whole document changes from, e.g., Computer Modern to Lucida Bright.

3 Normal usage

The vast majority of your existing LaTeX documents will run through the new system with no further change. The commonest area for incompatibilities is in math (see below, section 4), but you may also be 'tripped' by the new orthogonality in shape and weight selection. If your document contains:

```
See {\bf bold and then \tt typewriter}
```

under the old LaTeX, the `\tt` completely overrides the effect of `\bf`, but in the new system, they are different attributes, so the command `\tt` changes the *family* to (perhaps) `cmtt`, but the bold series attribute is unchanged, so the system will try to load a bold typewriter font (which you may or may not have). A rather more worrying situation occurs when you have defined a new command without thinking through the ramifications. Thus if in a style file or document preamble, we have said

```
\newcommand{\PS}{{\sc PostScript}}
```

so that `\PS` produces PostScript, and then later we write

```
\section{How \PS\ changed my life}
```

we may have a problem, if the `\section` command is defined to set its argument in a bold sans-serif typeface. The definition of `\PS` says that the word is in small caps. This is a change in *shape*, so the *family* (a sans-serif font) stays unchanged, as does the *series* (bold). So the system tries to load a small caps bold sans-serif font, which you may not have (you don't in Computer Modern, for instance). Here a very important feature of NFSS2 comes into action: *font substitution*. This follows a set of rules (which the user can change) to find the closest possible match. This may not be at all what the user intended, and it will probably produce a different effect from the old LaTeX. However, NFSS2 always puts out clear warnings on the terminal and log file when it is substituting, and the problem is usually obvious when you see your printout. Some more careful discipline is required when writing new documents, and emending old ones. Bear in mind that the strange effects are not *mistakes* by LaTeX, but simply a stricter interpretation of the input than the old LaTeX.

How do we persuade LaTeX to choose the attributes we want? The same commands which we already have work as expected, with two new ones available; these are listed in Table 1.

Command	*Meaning*	*Effect*
`\rm`	normal family	usually a serif font
`\sf`	sans family	a sans font
`\tt`	typewriter family	`a monospaced typewriter font`
`\bf`	bold series	**bold typeface**
`\it`	italic shape	*italic typeface*
`\sl`	slanted shape	*slanted typeface*
`\sc`	small-caps shape	SMALL-CAPS TYPEFACE
`\normalshape`	normal shape	back to normal
`\mediumseries`	normal series	normal weight

Table 1: User commands to change attributes

Command	*Example*	*Effect*
Changing *family*		
`\textrm`	`\textrm{Whales}`	Whales
`\textsf`	`\textsf{Whales}`	Whales
`\texttt`	`\texttt{Whales}`	`Whales`
Changing *series*		
`\textbf`	`\textbf{Whales}`	**Whales**
`\textmedium`	`\textmedium{Whales}`	Whales
Changing *shape*		
`\textit`	`\textit{Whales}`	*Whales*
`\textsl`	`\textsl{Whales}`	*Whales*
`\textsc`	`\textsc{Whales}`	WHALES
`\textnormal`	`\textnormal{Whales}`	Whales
`\textem`	`\textem{Whales}`	*Whales*

Table 2: New font commands with arguments

The size changing commands have an important difference compared to old LaTeX: they change *only* the font *size* attribute (in the old LaTeX they changed the series and shape back to 'normal roman').

The font changing commands in LaTeX declarations have always been rather anomalous, in that they affect the text within the `{` and `}` group where they are used, instead of having an argument like most other commands. LaTeX beginners often mistakenly type `\em{hello}` when trying to typeset *hello*. It is therefore a very good idea to start using a different set of commands, which are provided by the style option `fontcmds`, listed in Table 2.

If you want to change the *default* action associated with any of the above commands, you can do so with the `\renewcommand` macro; each of the declarations above has a corresponding command with a suffix of `default`. Thus you could change the effect produced by `\tt` by saying (in the document preamble or a style file):

```
\renewcommand{\ttdefault}{courier}
```

if you had a font family called 'courier'. There may be problems with the encoding, if this is a PostScript font, but we will discuss that in Appendix 1.

'How,' the suspicious reader will ask, 'do I know what values are allowed for the font attributes? How do I know that boldness is indicated by a series of "bx"?' In fact, more or less *any* value for the attributes is permitted, but if you want your document to be useable by others, it would be as well to stick to a conventional set. If you ask for a *shape* of 'grotesque', you will get the right font *if* the `fd` contains an entry for that combination of attributes. Conventional values are as follows (for the Computer Modern family):

Family cmr, cmss, cmtt

Shape n, it, sl, sc

Series m, b, bx (differentiating between normal bold and extended bold)

Most users will not worry about this, but simply use the high-level commands and get the effect they intended. Different font *families* will commonly be loaded via a style file which changes the default families looked for by `\rm`, `\sf` and `\tt`. A `palatino.sty`, for instance, will set things up so that the roman font is Palatino-Roman, the sans font is Helvetica and the typewriter font is Courier. A set of suitable `fd` files and style files for common PostScript fonts is distributed with NFSS2. The only problem here is agreeing on *family* names for fonts, and having suitable `fd` files, but this is done for a great many typefaces in the standard NFSS2 distribution, with the family names listed in Table 3.

4 Mathematical work

Mathematical typesetting with NFSS2 works in a very different way to text, as the fonts do *not* vary according to the current font attributes in the main body of the document. There are some incompatibilities with old LaTeX, and some new facilities. The principal difference is that font *declarations* like `\bf` no longer work, but we must rely on two different concepts, *math versions* and *math alphabets*. The

Computer Modern			
\cmr	Computer Modern Roman	\cmss	Computer Modern Sans
\cmtt	Computer Modern Typewriter	\cmm	Computer Modern math italic
\cmex	Computer Modern math extension	\cmsy	Computer Modern math symbols
\cmdh	Computer Modern Dunhill	\cmfib	Computer Modern Fibonacci
\cmfr	Computer Modern Funny		
LaTeX			
\lasy	LaTeX symbols		
AMS			
\msa	AMS symbol font 1	\msb	AMS symbol font 2
Concrete			
\ccmi	Concrete math italic	\ccr	Concrete Roman
Euler			
\euex	Euler math extension	\euf	Euler Fraktur
\eur	Euler Roman	\eus	Euler Script
PostScript			
\pag	Avant Garde	\pbk	Bookman
\pcr	Courier	\phv	Helvetica
\ppl	Palatino	\psy	Symbol
\ptm	Times	\pzc	Zapf Chancery
\pzd	Zapf Dingbats		

Table 3: Common font family names

mathversion changes the appearance of the whole formula (all the fonts change), while the *alphabet* is used to set a particular set of of characters in a chosen font. The normal text commands like `\rm`, `\sf` or `\bf` are now completely illegal in math, and a new set of commands is provided:

Example	*Effect*
`\mathcal`	calligraphic style
`\mathrm`	upright text
`\mathbf`	bold text
`\mathsf`	sans-serif
`\mathit`	italic text

These are *commands* with an argument, not declarations. So to get a calligraphic ABC, we say `$\mathcal{ABC}$`, and *not* `$\mathcal ABC$`. The effect of the latter will be to set just the A in calligraphic, since just the first token after `\mathcal` is taken as the argument.

You can define *new* math alphabets for yourself easily, with the `\DeclareMathAlphabet` command, which associates a particular font family, encoding, shape, and series with the command you want to use. So if we want to declare a typewriter math alphabet, we could say (in the document preamble):

```
\DeclareMathAlphabet
        {\mathtt}{OT1}{cmtt}{m}{n}
```

(`OT1` is the name of the original TeX font layout).

What about new math symbol fonts? To illustrate some of the commands available here, let us look at how a style file looks which sets up the AMS symbol fonts. Assuming that the relevant `fd` files exist on our system for the fonts (named *msa* and *msb* in Table 3 above), we can declare the existence of them as symbol fonts:

```
\DeclareSymbolFont{AMSa}{U}{msa}{m}{n}
\DeclareSymbolFont{AMSb}{U}{msb}{m}{n}
```

where we define the names (`AMSa` and `AMSb`) by which we are going refer to them in future, the *encoding* (U is for 'undefined', where there is no standard layout), *family* (msa and msb), *series* (m) and *shape* (n). We can use the new fonts in two ways:

1. By declaring named math symbols, e.g.:

   ```
   \DeclareMathSymbol\lozenge
           {\mathord}{AMSa}{"06}
   ```

   which looks more fearsome than it really is. We are defining a new math macro `\lozenge`, and saying it comes from the `AMSa` font we have defined earlier, at position `"06` (this hexadecimal numbering notation is described in 4, p. 116). The tricky bit is `\mathord`, which says what *type* of symbol it is. The possibilities are listed in Table 4, but the serious user is advised to consult 3, p. 158.[2]

2. By saying that we want to use a symbol font also as a *math alphabet*:

   ```
   \DeclareSymbolFontAlphabet
           {\bbold}{AMSb}
   ```

[2] Note that `\mathalpha` is an NFSS2 addition to the list of possible types.

This defines a new *math alphabet* command `\bbold`, which picks up characters from the `AMSb` fonts (where the AMS has placed 'blackboard bold' letters).

The use of the three math alphabet and symbol commands here is used to construct most of the standard math interface; the supplied style file `euler.sty`, which redefines math to use the Euler fonts, is a good example of its use.

I have not dealt much here with *math versions*; suffice it to say that each of the commands described above has a corresponding command which allows the user to select a specific symbol or alphabet font for each different math version.

5 Going further, and towards LaTeX3

This article has not described all the features of NFSS2, or the style files which are distributed with it. For a detailed description of the whole system (and a great many other useful topics in LaTeX), the reader is referred to 1.

I advise all LaTeX users to switch to the new NFSS2 standard for LaTeX as soon as possible; the work done in NFSS2 is central to the development of the next version of LaTeX, written by the official maintainers of LaTeX, and if you get used to the NFSS way of thinking now, you will be in a good position to take advantage of LaTeX3 when it appears. The few remaining common style files which rely on the old behaviour of LaTeX are rapidly being converted, and many new font-related styles are being written now that it is so much easier to do so.

References

[1] Goossens, M., Mittelbach, F., and Samarin, A. (1993). *The LaTeX Companion.* Addison-Wesley, Reading, Massachusetts.

[2] Haralambous, Y. (1992). TeX conventions concerning languages. *TeX and TUG News*, 1(4):3–10.

[3] Knuth, D. E. (1984). *The TeXbook.* Addison-Wesley, Reading, Massachusetts.

[4] Lamport, L. (1986). *LaTeX Users Guide and Reference Manual.* Addison-Wesley, Reading, Massachusetts.

Appendix 1: Encoding, and PostScript fonts

It is an unfortunate fact of life that we do not have a completely accepted standard for layout of fonts; everyone agrees on a certain number of characters, and where they are to be found (the ASCII character set), but this is not adequate for any serious typesetting. Whenever we type character 234 or a TeX macro like `\c{c}`, we want the effect to be ç, and TeX has to know where to find this in a font. We are likely to meet at least three situations, or five if we use PostScript fonts:

1. The layout of Knuth's original Computer Modern fonts (3, Appendix F).
2. The 'Cork' layout, discussed by 2.
3. An extended ASCII layout, as used (for instance) in many DOS or Macintosh applications (not necessarily the same!).
4. 'Raw' PostScript layout, the default defined by Adobe.
5. A re-coded PostScript layout; an example is in the font metric files supplied with Tom Rokicki's *dvips*, which is basically the same as Knuth's layout, but has some small differences.

This means that the definition of the `\c` macro is going to vary according to the font we are using; hence the font attribute of *encoding.*[3] NFSS2 provides *hooks*, so that we can supply TeX code which will be activated when the encoding changes. Alternatively, if we know we are going to use a different encoding throughout a document, we can just make changes in a style file.

The normal user will have to make one (important) choice: is she going to use the current Computer Modern Roman layout, or the new Cork layout? If she chooses the latter (and this means the 'dc' fonts must be available), then the style file `dclfont.sty` is supplied to set up all the necessary macros.[4] If use of PostScript fonts is proposed, slight complications ensue. In the first case (original TeX layout), then the PostScript fonts must be in exactly the right layout; users of Rokicki's metric files and *dvips* will need to use the style file `psnfss.sty`, and use the command `\RokickiExtras` in the document preamble to fix some macros. In the case of using Cork encoding, it will be necessary to get metrics for the PostScript fonts which remap the layout correctly, and to ensure that the PostScript interpreter in the printer knows about it too. This is discussed in detail in the *PSNFSS2* sub-package supplied with NFSS2.

Assuming the encoding question has been sorted out, the important thing is to change all the family defaults. Setting up a document to use (say) PostScript Times Roman throughout is trivial; we simply write:

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
```

[3] This is an addition to the NFSS since version 1.

[4] This will become the *default* in LaTeX3.

Type	*Meaning*	*Type*	*Meaning*
`\mathord`	ordinary	`\mathop`	large operator
`\mathbin`	binary operator	`\mathrel`	relation
`\mathopen`	opening	`\mathclose`	closing
`\mathpunct`	punctuation	`\mathalpha`	alphabetic

Table 4: Math symbol types

in a style file or in the document preamble, and we will have changed the standard for normal, sans-serif and typewriter setting.

Some users will not be working with anything like the ASCII layout at all, but will be typesetting in something like Cyrillic, Devanagari or Elvish. The *encoding* attribute for fonts makes it easy to swap between different systems in the same document, and call up whatever macros are needed for a particular situation. New encodings can be defined, and old ones redefined. The detailed commands for declaring new encodings, and for declaring new font families, are described in the NFSS2 documentation.

Appendix 2: Installation

This section is relevant to the reader only if NFSS2 has not already been installed on her computer.

One of the characteristics of the new work being undertaken on LaTeX is that Knuth's principles of literate programming are being applied, which means that a single documented source is supplied from which the user can either generate printed documentation, or produce a useable file for TeX input. The NFSS2 distribution cannot therefore be used until it has been unpacked and installed; this is all done using TeX itself, and the only additional package needed by the first time user is the *docstrip* macros, which should always be supplied with NFSS2. The installation is in two parts:

1. We must first get a set of `fd` (and some LaTeX `.sty`) files ready for use; this is done by running plain TeX or LaTeX on the file `main.ins` which generates all the files we need. It will probably ask some questions as it runs about what fonts you have, but it won't matter much if you get the answers wrong (it may create `fd` files for strange fonts you don't have, but if you never try to use them, that does not matter).
2. Now we need to create a new LaTeX *format file* (usually with a suffix of `.fmt`); this can seem a slightly forbidding procedure, but should be explained in the documentation with your TeX; what happens is that a special version of TeX is run, called iniTeX, which reads the basic LaTeX macros, and hyphenation patterns, and dumps a fast loading version which you place where normal TeX can find it. On some systems there is actually an `initex` command, but on others it is an option to normal TeX. Thus, if you use the emTeX package, you type `tex -i` to use the right portion of the program. You *must* use this special TeX, or you cannot use the hyphenation system.

   Creating the NFSS2 format is straightforward—just run iniTeX on the file `nfss2ltx`, and type `\dump` in response to the `*` prompt when it finishes loading files.[5]

If you want to test your new LaTeX format now, work in the directory where you placed the NFSS2 files; otherwise move `nfss2ltx.fmt` to the directory where your TeX looks for format files, and copy all the `fd` files to a directory which TeX searches for inputs (along with any `.sty` files created in stage 1 above). Depending on how your TeX system works, you may have to do some more work to access the format file; thus we might edit the `latex.bat` file[6] under DOS, or make a new symbolic link to `virtex` if we run the *web2c* Unix TeX.

If you wish to find out more about NFSS2, and maybe print the documentation of its internal workings (not for the faint of heart!), read the file `readme.mz8` for details of how to proceed.

◇ Sebastian Rahtz
ArchaeoInformatica
12 Cygnet Street
York Y01 2JR
UK
spqr@minster.york.ac.uk

[5] Unless you are a rather advanced guru, and commonly add extra features to your format files.

[6] It is strongly advised that you do *not* make your new LaTeX an option, by making a new command, but that you use it for all your work from now on.