nfssext-cfr

Clea F. Rees*

v1.0 (SVN Rev: 10366) 2024/09/18

Abstract

nfssext-cfr is an extension and modification of Philipp Lehman's nfssext which provides extended font selection commands modelled on those provided by LaTeX 2ε. Given an appropriate font configuration, nfssext-cfr enables users to change the weight, width, shape and style of font as easily as they can select bold, italic or typewriter. For instance, the package makes it trivial to use proportional, hanging figures in the body of the text, proportional, lining figures in captions and headers and tabular, lining figures in tables. An extensive choice of commands are provided to access a wide variety of weights, widths, shapes and styles from the more common (e.g. semi-bold or condensed) to the less common (e.g. 'outline' and right or upright italic). Comprehensive support is provided for 'swash' and 'alternate' styles. These are implemented as families rather than shapes to support fonts which offer multiple swash shapes (e.g. small-caps, italic and upright) or alternate styles. These may be used to provide effective access to fancy ligatures, end-of-word swashes etc. without sacrificing the range of characters provided by T1.

The package is not primarily intended for direct use by end-users, but is designed rather to facilitate the creation of more sophisticated font support packages. End-users may nonetheless find the package useful, subject to the constraints explained in this document. Moreover, end-users may wish to pass options to the package on newer kernels, if loading font support packages which have not been updated for changes to font selection[1].

Contents

*Bug tracker: codeberg.org/cfr/nfssext/issues | Code: codeberg.org/cfr/nfssext | Mirror: github.com/cfr42/nfssext

[1] 'Kernel' refers to the LaTeX kernel in this context and should not be confused with your system kernel.

List of Tables

1 Introduction

The package was originally a fairly simple extension of Philipp Lehman's nfssext. nfssext provides commands which enable one to specify font features not covered by the New Font Selection Scheme (NFSS). The package developed according to the needs of particular fonts I configured for LaTeX and, in a few cases, my dissatisfaction with the original commands.

In adapting the package for the (New) New Font Selection Scheme[2] (NNFSS), I have tried to balance (i) backwards compatibility[3] for users loading updated font support packages, (ii) backwards compatibility for users loading packages which haven't been updated and (iii) compatibility with the new features of NNFSS. I have also tried to account for the common case in which documents use combinations of fonts from different packages, each of which may or may not load nfssext-cfr and may or may not have been updated for NNFSS. This has inevitably required some compromises and there are certainly places where I would do things differently if starting from scratch.

While I don't recommend installing this version of nfssext-cfr on an older system, the package should continue to work more-or-less as it always did on older kernels. To achieve this, the package is split into a main file, `nfssext-cfr.sty`, which provides common code and figures out whether to load code for NFSS (`nfssext-cfr-nfss.sty`) or NNFSS (`nfssext-sty-nnfss.sty`).

[2] Officially, there is no such designation, but I have to call it something. Throughout this document and packages which depend upon it, I use this term to refer to the font selection features introduced into LaTeX in 2020.

[3] Note that 100% backwards compatibility cannot be implemented on current LaTeX kernels.

`nfssext-cfr-nfss.sty` is essentially what was `nfssext-cfr.sty` minus the code retained in the latter.

Unfortunately, tt is impossible to ensure 100% backwards compatibility with recent LaTeX kernels. By default, nfssext-cfr tries to interfere as little as possible with the kernel, even at the cost of backwards compatibility. If `compat` is enabled, however, the package does its best to enable backwards compatible behaviour, at the cost of the new functionality provided by the kernel.

nfssext-cfr does not eschew interference with the kernel at all costs. Even without `compat` it patches or replaces some kernel code because some things just don't work sensibly[4].

All font-support code should be updated to use `compat=false` when loading nfssext-cfr. The changes in NNFSS **require** changes to code based on Philipp Lehman's Font Installation Guide.

2 Macros

Tables 1 to 7 include macros supplied by the original nfssext and additions available with nfssext-cfr. Macros in tables 5 to 7 should work with any font definition files which more-or-less adhere to NFSS/NNFSS. This should, theoretically, be all font packages but, in practice, things are rarely so simple. Macros in tables 3 and 4 will work only with fonts named strictly according to the Berry naming scheme.

In tables 3 to 7, the third column lists the default letter codes for various font features. If the defaults are changed, the macros will try to do something different.

A `+` indicates that the macro will attempt to merge the addition into the current font's family name, series or shape. For example, if the current font uses oldstyle figures, the `+2` indicates that `\pstyle` will attempt to select a font with figures which are both proportional and oldstyle.

A `-` indicates that the macro will attempt to subtract from the current font's family name, series or shape. For example, if the current font uses oldstyle figures, the `-2` indicates that `\tstyle` will attempt to select a font with figures which are both tabular and oldstyle.

A comma-separated list indicates consecutive additions and/or subtraction.

If no `+-` is used, the macro tries to select a font with the given feature without merging. For example `\sistyle` tries to switch to `si` shape regardless of the current font shape.

A `-` indicates that the macro will try to clear all relevant letter codes from the current font's family name, series or shape. For example, `\regwidth` tries to switch

[4] This is true in two main places. The first is the kernel's initialisation of series at the start of the document. This overwrites the default `bf` series according to the font family name rather than the font name. This means that virtual fonts which depend on Computer or Latin Modern are not handled correctly and, because this code is delayed, the problem cannot be corrected by setting things up appropriately earlier.
The second is the implementation of 'swash' which is by far the most problematic of the changes and one of the most difficult to navigate. It isn't clear to me how seriously the kernel's definition is intended to be, but I have chosen to overwrite the kernel code here.

Table 1: Standard (kernel) macros (re)defined

`\swshape`[a,b]	redefined on new kernels[c] defined on old kernels
`\itshape`	old kernels only
`\scshape`	old kernels only
`\upshape`	old kernels only

[a] Defined only by newer kernels.
[b] See tables 3 and 5 and text.
[c] Definition depends on kernel, `force` and `compat`.

Table 2: Standard (kernel) font change rules redefined

Shape		
Current	Requested	Applied when?
`it`	`sc`	`compat` & NNFSS only
`sl`	`sc`	`compat` & NNFSS only
`sc`	`it`	`compat` & NNFSS only
`sc`	`sl`	`compat` & NNFSS only
`scsl`	`it`	`compat` & NNFSS only

to a series with no letter codes indicating non-standard widths in its name.

Additions, subtractions and clearances operate on font family names, series or shapes, as appropriate. In general, macros with `style` in their names operate on family names; those with `shape` operate on shape codes[5]; and those with `width` or `weight` operate on series codes.

The letter codes correspond to those specified by the NFSS specification, unless the specification does not include the relevant feature. In the latter case, I tried to choose something sensible i.e. something which made sense to me at the time. These choices are not always those specified by the NNFSS specification, since sense and sensibility are sometimes in the eye of the encoder.

One further macro is available, though it has no effect on older kernels.

`\nfssextset` {⟨*key-value list*⟩}

Package options (see section 3 may also be specified after loading either in the preamble (`compat` and `force`) or at any time (`debug`). This enables users to set options after some other package loads nfssext-cfr and allows additional information to be printed to the console and/or logged on local basis.

[5] But **not** `\swshape`!

Table 3: Family switches: general

Macros			
Text Command	Switch	Family Code	Style
`\textti`	`\tistyle`	`+d`	titling/display
`\textlt`[a]	`\ltstyle`[b]	`+l`	light if separate family
`\textof`	`\ofstyle`	`+l`	open-face (or outline or blank) style
`\textalt`	`\altstyle`	`+a`	alternative style
`\textreg`	`\regstyle`	`-`	regular style
`\emboss`	`\embossstyle`	`+e`	‘embossed’ style
`\textorn` `\ornament`	`\ornamentalstyle`	`+p`	decorative initials etc.
`\textqt`	`\qtstyle`	`+q`	quotation style
`\textsh`	`\shstyle`	`+h`	shadowed style
`\texttm`	`\tmstyle`	`-s,-v,+t`	monowidth typewriter
`\texttv`	`\tvstyle`	`-s,-t,+v`	variable width typewriter
`\textswash`	`\swashstyle`[c]	`+w`	swash
`\textsw`[d]	`\swshape`[d]		‘find a route to swash’

[a] Cf. `\textlg` in table 7.

[b] Cf. `\lgweight` in table 7.

[c] Cf. `\swstyle` in table 5.

[d] Effect is kernel and option dependent, but potentially changes family and/or shape. ‘Tries to find a route to swash.’ See text for an explanation of what, why and when. See section 6 for details of how.

Table 4: Family switches: figures

Macros			
Text Command	Switch	Family Code	Style of Figures
`\textln`[a]	`\lnstyle`[a]	`-`	lining (cf. `\lstyle` below)
`\textos`[a]	`\osstyle`[a]	`j`	oldstyle (cf. `\ostyle` below)
`\textinf`	`\infstyle` `\instyle`	`0`	inferior
`\textin`[b]			if hyperref is not loaded
`\textsu`	`\sustyle`	`1`	superior
`\textl`[c]	`\lstyle`[c]	`-j`	lining (cf. `\lnstyle` above)
`\texto`[c]	`\ostyle`[c]	`+j`	oldstyle (cf. `\osstyle` above)
`\textp`[c]	`\pstyle`[c]	`+2`	proportional
`\textt`[c]	`\tstyle`[c]	`-2`	tabular
`\textpl`[d]	`\plstyle`[d]	`-j,+2`	proportional lining
`\textpo`[d]	`\postyle`[d]	`+2j`	proportional oldstyle
`\texttl`[d]	`\tlstyle`[d]	`-j,-2`	tabular lining
`\textto`[d]	`\tostyle`[d]	`+j,-2`	tabular oldstyle

[a] This macro is the original nfssext command. The result is independent of the current style of figures.

[b] Deprecated.

[c] This macro changes precisely one aspect of the current figure style. That is, the result depends on the current style of figures.

[d] This macro ensures a specific fully-specified figure style.

Table 5: Shape switches

Macros			
Text Command	Switch	Shape Code	Shape
–	`\scolshape`	`scol`	outline small-caps
`\textol`	`\olshape`	`ol`	outline
`\textsi`	`\sishape`	`si`	italic small-caps
`\textu`	`\ushape`	`u`	??
`\textscu`	`\scushape`	`su`	??
`\textui`	`\uishape`	`ui`	upright italic
`\textri`	`\rishape`	`ri`	reverse italic
`\textdf`	`\dfshape`	`n`	default shape
–	`\swstyle`[b]	`+w,it`	swash family *and* shape
`\textsw`[a]	`\swshape`[c]		‘find a route to swash’

[a] Cf. `\textswash` in table 3.

[b] Cf. `\swashstyle` in table 3.

[c] Definition is kernel and option dependent, but probably doesn’t (just) change shape. See table 3 for sketch and text for details.

Table 6: Series switches: widths

Macros			
Text Command	Switch	Series Code	Width
`\textnw`	`\nwwidth`	`+c`	narrow
`\textcd`	`\cdwidth`	`+c`	compact
`\textec`	`\ecwidth`	`+ec`	extra compact
`\textuc`	`\ucwidth`	`+uc`	ultra compact
–	`\mdwidth`	`+?m`	medium
`\textet`	`\etwidth`	`+x`	extended
`\textep`	`\epwidth`	`+x`	expanded
`\textex`	`\exwidth`	`+ex`	extra expanded
`\textux`	`\uxwidth`	`+ux`	ultra expanded
`\textrw`	`\regwidth`	`-`	regular

Table 7: Series switches: weights

Macros			
Text Command	Switch	Series Code	Weight
–	`\mdweight`	`+m?`	medium
`\textmb`	`\mbweight`	`+mb`	medium-bold
`\textdb`	`\dbweight`	`+db`	demi-bold
`\textsb`	`\sbweight`	`+sb`	semi-bold
`\textbd`	`\bdweight`	`+b`	bold
	`\bfweight`		
`\texteb`	`\ebweight`	`+eb`	extra-bold
`\textub`	`\ubweight`	`+ub`	ultra-bold
`\textlg`[a]	`\lgweight`[b]	`+l`	light when weight
`\textel`	`\elweight`	`+el`	extra-light
`\textul`	`\ulweight`	`+ul`	ultra-light

[a] Cf. `\textlt` in table 3.

[b] Cf. `\ltstyle` in table 3.

3 Newer LaTeX Kernels

The package tests for the presence of `\init@series@setup`. If this exists, it loads a newer version of the package. nfssext-cfr supports three options, but these are only effective if the newer code (for NNFSS) is loaded. All three are booleans, initially false and default to true if used without specifying a value[6]. The third option is described in section 5.

`force` (*opt.*) `=true|false`

Default: `true`

Initial: `false`

Scope: preamble

You can force the old code to be loaded using the package option `force` or `force=true`. Note, however, that the old version will not work as advertised on newer kernels because LaTeX will overwrite some of the package's definitions at the end of the preamble.

`compat` (*opt.*) `=true|false`

Default: `true`

Initial: `false`

Scope: preamble

In contrast, `compat` or `compat=true` will activate code which tries to partially replicate the original nfssext-cfr's behaviour. This is far from unproblematic. In particular, it will partly break features of the current NFSS for other fonts.

If your document relies exclusively on text fonts supported by this package and none of the support for those fonts has been updated, compilation should produce a more satisfactory result than otherwise. If, however, your document relies partly on text fonts not supported by this package or the support for those fonts has been updated in the 'wrong' way, compilation may produce a less satisfactory result. There is no general rule here: whether the option helps or hinders things depends entirely on the fonts, the support for those fonts and the specific contents of your document.

The main areas known to be problematic are

1. italic small-capitals (but oblique small-capitals should be mostly unaffected);
2. swash;
3. transitions between small-caps, italic, oblique, italic small-caps, oblique small-caps, upright italic, right italic and upright;
4. any transition involving swash where shape is involved;

[6] 'Default' and 'initial' follow the usage in `l3interface.pdf`, l3keys2e and `clsguide.pdf`. If you are familiar with pgfkeys, the terms have the same meaning there. Basically, the 'initial' value is what you get if you don't specify the option at all when loading the package, while the 'default' is what you get if you specify the option without specifying a value.
'Scope' is used in the standard sense applicable to LaTeX 2_ε class and package options. That is, it indicates whether the option may be used only when loading the package, at any point in the preamble or also in the document.

5. medium weight fonts where width is non-standard e.g. medium condensed, medium ultra condensed etc.;

6. medium bold weight in any context.

1–4 can be worked around at the document level, with some inconvenience. Subject to the caveats above, the `compat` option may avoid at least some of these inconveniences.

5 and 6 cannot be worked around at the document level. Nor does nfssext-cfr make any attempt to mitigate these two issues. Doing so would involve too much interference with current NNFSS. This means that 5 and 6 can be addressed only in the support files for the fonts affected. Neither `compat` nor `force` makes any attempt to change this.

Maximum backwards compatibility requires changes to the font support files *and* `compat=false`, but some documents may still require (hopefully minor) changes.

3.1 Required Changes to Font Support Files

In all cases, additional changes to font substitution rules may be needed to prevent multiple substitutions by the same font, since these seem to cause problems.

italic small-caps The problem here is that nfssext encoded italic small-caps as shape `si`, whereas the kernel has plumped for `scit`. It does support `scsl` (although it does not distinguish oblique from italic), but not `si`. This issue can be more-or-less dealt with by support files for fonts, but some issues arise at the document level concerning transitions (below).

Ideally, `scit` should be substituted wherever font definition files specify the shape `si`. `scit` should then be defined as a (silent) substitution for `si`. However, it *should* be sufficient to define `si` as a substitute for `scit`.

Fonts which provide oblique small-caps, but not italic, should specify `scsl` as a (silent) substitution for `scit` and `scit` for `si` (or *vice-versa*). The kernel supports `scsl` out-of-the-box, together with the substitution for `scit`, but the changes should make support for `\textsi` and `\sishape` more robust.

transitions No additional changes are recommended to support files which load nfssext-cfr. Provided `fd` files are updated as explained above, no further adjustments should be required to enable correct font selection during transitions involving italic, oblique, small-caps, upright italic, reverse italic etc.

swash No changes are recommended for swash to font definition files for packages which load nfssext-cfr. If loading nfssext-cfr, the recommendation is to **ignore** the kernel's implementation because it cannot be made to work correctly with any family which provides swash for multiple shapes e.g. both upright and italic or small-caps and upper/lower case[7].

[7] The issue here is that the kernel considers swash to be a *shape*, whereas nfssext only *called* it a shape. The underlying code treated it as a *style* requiring a change of font *family*. The shape

swshape No changes for swash are recommended for `sty` files in packages which load nfssext-cfr.

nfssext-cfr-nfss contains the original nfssext definition of `\swshape` and `\textsw`. This is used on newer kernels only if `force` is used, in which case the code is largely broken.

nfssext-cfr-nnfss contains both the original definition and a replacement. The former is used only if `compat` is selected; otherwise, the latter is used on kernels supporting NNFSS. The new definition tries to figure out which of the three possible implementations, if any, to use and behaves accordingly. **The kernel definition is overwritten regardless.** `compat` determines only *what* overwrites the kernel's.

medium Any line of a font definition file which codes a series of two or more letters including `m` must be changed to delete the `m`. For example, `{mc}` → `{c}`, `{muc}` → `{uc}` etc. **It is NOT sufficient to substitute such series using rules.** The changes **must** be made in the primary definitions of the font families.

mb I've chosen to make `\mbweight` an alias for `\sbweight`. Hopefully no font family supports both. Provided that's not the case, `mb` should be changed to `\sb` in all affected font definition files. **It is NOT sufficient to substitute `mb` for `sb` using a rule.** The change **must** be made in the definition of the family.

4 Older LaTeX Kernels

This is the code base nfssext was written for. nfssext-cfr extended that code.

To the best of my knowledge, the code used with older LaTeX kernels works as expected. This means it is fully compatible with the Font Installation Guide and that things like `\scshape \itshape` will produce italic small-caps, as expected. This code is also — again to the best of my knowledge — fully compatible with all features of NFSS with the single exception of code supporting medium weight, condensed width fonts which erroneously uses `mc` rather than `c`.

Italic small-caps is assumed to be coded as `si`. Oblique small-caps is assumed to be coded as `scsl`.

If a set of fonts provides a swash *family*, it is assumed the fonts are named in accordance with the Berry scheme. `\textswash`, `\swshape`, `\swashstyle` and/or `\swstyle` can then be used to access this family.

The difference between `\swashstyle` and `\swstyle` is that the former tries to merge any swash family with the current one, whereas the latter does not. So, if

was always (potentially switched to) italic (`it`). nfssext-cfr offered a second version of swash, which treats it as a family possibly requiring a change of shape, but the shape is typically italic or upright, as opposed to being specific to swash.

To make things worse, not all fonts *can* be setup in the way the kernel assumes because some fonts provide swash characters in a variety of shapes (upright and italic, for example). Moreover, it is common to encode additional ligatures, for example, as swash, even though this is not accurate, in order to provide *some* mechanism for accessing them within a traditional 8-bit font setup.

a font set provides swash for two widths of font, say, and you've changed widths, `\swashstyle` will try to find a swash character without altering the width, whereas `\swstyle` will first switch to the base font, resetting the width. On the other hand, if the font only provides swash in the standard width, say, and you've changed widths, `\swashstyle` will fail to switch to swash, whereas `\swstyle` will succeed. Packages which include swash families should, therefore, advise users which command(s) to use.

`\textswash` is the text font command for `\swashstyle`. `\textsw` is the text font command for `\swshape`.

`\swshape` first tries `\swstyle` before changing the shape to `\swshapedefault`. By default, this is `\itdefault` because swash families are often coded as italic, but this is obviously font-dependant.

5 Bugs, Non-Bugs & Debugging

The actual effect of any macro depends on any changes made to the defaults for various font features, the current font and, of course, what is available.

The macros operating on family names are almost entirely reliant on font names adhering strictly to the Karl Berry schema. This includes the stipulation that multiple variants be listed in alphabetical order. These macros cannot be used with fonts named in any other way.

On older kernels (NFSS), changes to weight and width should work and most shapes should be supported, but italic small-caps is assumed to be coded as `si` on these kernels, so you may need something like

```
\renewcommand*{\sidefault}{scit}% or scsl or whatever
```

On newer kernels, italic small-caps should be encoded as explained above and `\sishape`/`\textsi` should work out-of-the-box with packages which use any of `si`, `scit` or `scsl`.

If a macro's attempt to enable or disable a font feature fails, a warning will generally be written to the console, but the code tries hard not to trigger errors. If an attempt triggers an error, that's a bug, so please let me know. If an attempt triggers a warning, please note that there may be no bug at all and, if there is a bug, it is probably not in this package[8]

`debug` (*opt.*) `=true|false`

Default: `true`

Initial: `false`

Scope: general

You can get a bit more information printed to the console about what's happening using this option on newer kernels. Some of the same information can also be

[8] To be clear, there certainly are bugs. It is just statistically unlikely that any given warning is caused by one.

found in the log.

6 Implementation

You do not need to read the remainder of this document in order to install or use the package.

6.1 Main package file

This used to be the complete package. Now it is mostly responsible for processing options, figuring out which kernel we're on and implementing the small amount of code shared between the implementations for NFSS and NNFSS.

`nfssext-cfr` (*pkg.*)

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{svn-prov}
\ProvidesPackageSVN[\filebase.sty]{$Id: nfssext-cfr.dtx 10366 2024-09-18 14:25:21Z cfrees $}[v1.0 \revinfo{} extensions for NFSS and NNFSS; based on 2003/03/14 v1.2 Experimental NFSS Extensions]
\DefineFileInfoSVN
```

`\if@nfssextcfr@digonnew`

```
\newif\if@nfssextcfr@digonnew
```

```
\@nfssextcfr@digonnewtrue
```

Copied verbatim, excepting format and modulo package/module name from Joseph Wright's `siunitx.sty` under LPPL

```
\@ifundefined{ExplLoaderFileDate}{%
  \IfFileExists{expl3.sty}{%
    \RequirePackage{expl3}%
  }{%
    \RequirePackage{nfssext-cfr-nfss}%
    \@nfssextcfr@digonnewfalse
  }%
}{}
\if@nfssextcfr@digonnew
```

Almost verbatim from `siunitx.sty`

```
  \@ifl@t@r\ExplLoaderFileDate{2022-02-24}{%
  }{%
    \RequirePackage{nfssext-cfr-nfss}%
    \@nfssextcfr@digonnewfalse
  }%
\fi
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\if@nfssextcfr@digonnew
```

BEGIN expl pkg option setup

```
\newif\ifexfs@debug
\ExplSyntaxOn
\prop_gput:Nnn \g_msg_module_name_prop { nfssext-cfr } { exfs }
\keys_define:nn { exfs }
{
```

`compat` (*opt.*)
`\g__exfs_compat_bool`

Compatibility option.

```
  compat .bool_gset:N = \g__exfs_compat_bool,
  compat .default:n = true,
  compat .initial:n = false,
  compat .usage:n = preamble,
```

`debug` (*opt.*)

Turns info messages into warnings for testing purposes and possibly provides additional information.

```
  debug .legacy_if_gset:n = exfs@debug,
  debug .initial:n = false,
  debug .default:n = true,
```

`force` (*opt.*)
`\g__exfs_force_bool`

Force loading of code for NFSS even on newer kernels.

```
  force .bool_gset:N = \g__exfs_force_bool,
  force .default:n = true,
  force .initial:n = false,
  force .usage:n = preamble,
```

```
}
```

Joseph Wright: from `siunitx.sty` ; https://chat.stackexchange.com/transcript/message/64327823#64327823

```
\providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }
\IfFormatAtLeastTF { 2022-06-01 }
{
  \ProcessKeyOptions [ exfs ]
}{
  \RequirePackage { l3keys2e }
  \ProcessKeysOptions { exfs }
```

`\ProcessKeyOptions`

```
  \NewDocumentCommand \ProcessKeyOptions { o }
  {
    \IfValueTF { #1 } { \ProcessKeysOptions { #1 }  }
    { \PackageError{nfssext-cfr}{
        Optional~argument~mandatory~on~kernels~this~old.\MessageBreak
        Please~specify~the~module~whose~keys~should~be~processed
      }{
        \protect\ProcessKeyOptions~only~passes~keys~to~
        \protect\ProcessKeyOptions\MessageBreak
        on~older~kernels~for~the~convenience~of~package~authors.~
        Since~\protect\ProcessKeysOptions\MessageBreak
        takes~an~argument,~the~optional~argument~to~
        \protect\ProcessKeyOptions~is~required
      }
    }
  }
```

```
}
\IfFormatAtLeastTF { 2020-10-01 }{
}{
  \RequirePackage { xparse }
```

\ExpandArgs

```
  \providecommand \ExpandArgs [1]
  { \cs_if_exist_use:c { exp_args:N #1 } }
```

```
}
```

END expl pkg option setup

BEGIN cfr-added: bifurcate

We test for the presence of `\init@series@setup` in order to determine whether to load code for NNFSS or NFSS. If `force` is set, we load for NFSS regardless.

```
\RequirePackage{etoolbox}
\msg_new:nnn { nfssext-cfr } { compat }
{
  You ~ or ~ a ~ font-support ~ package ~ have ~ loaded ~ me ~ ( line ~ \msg_line_number: ) ~ with ~ the ~ compat ~ option. ~
  This ~ means ~ the ~ package ~ may ~ require ~ updating. ~
  Update ~ the ~ package ~ if ~ necessary ~ and ~ use ~ compat=false ~ when ~ loading ~ me.
}
\msg_new:nnn { nfssext-cfr } { force }
{
  You ~ or ~ a ~ font-support ~ package ~ have ~ loaded ~ me ~ ( line ~ \msg_line_number: ) ~ with ~ the ~ force ~ option. ~
  This ~ is ~ a ~ desperate ~ measure ~ of ~ last ~ resort. ~
  **Breakage ~ is ~ expected.**
}
\hook_gput_code:nnn { begindocument/before } { . }
{
  \cs_if_exist:NTF \init@series@setup
  {
    \bool_if:NTF \g__exfs_force_bool
    {
      \msg_warning:nn { nfssext-cfr } { force }
      \RequirePackage {nfssext-cfr-nfss}
    }{
      \bool_if:NT \g__exfs_compat_bool
      {
        \msg_warning:nn { nfssext-cfr } { compat }
      }
      \RequirePackage {nfssext-cfr-nnfss}
    }
  }{% achosiad heb broblemau - diolch byth!
    \RequirePackage {nfssext-cfr-nfss}
  }
}
```

__exfs_set:n

```
\cs_new_protected_nopar:Nn \__exfs_set:n
{
  \keys_set:nn { exfs } { #1 }
}
```

\nfssextset Allow setting of options later in preamble or in document. This is intended to enable debugging to be toggled locally.

```
\cs_set_eq:NN \nfssextset \__exfs_set:n
```

```
\ExplSyntaxOff
```

END added

\DeclareTextOrnament \ornament These are unmodified from nfssext. I'm not aware of any CTAN packages using these and they have not been tested for compatibility with NNFSS, though I can't see any patently obvious problems.

```
\newcommand*{\DeclareTextOrnament}[7]{%
  \expandafter\def\csname#1@orn\@roman#2\endcsname{#3/#4/#5/#6/#7}}
\begingroup
  \catcode`\/=12
  \gdef\exfs@split@orndef#1/#2/#3/#4/#5\@nil{%
    \def\f@encoding{#1}%
    \def\f@family{#2}%
    \def\f@series{#3}%
    \def\f@shape{#4}%
    \def\exfs@tempa{#5}}
\endgroup
\def\exfs@base@family{\expandafter\exfs@get@base\f@family\@nil}
\DeclareRobustCommand{\ornament}[1]{%
  \expandafter\ifx\csname\exfs@base@family @orn\@roman#1\endcsname\relax
    \PackageWarning{nfssext}{%
      Ornament #1 undefined for font family '\exfs@base@family'\MessageBreak
      Setting debug mark}%
    \rule{1ex}{1ex}%
  \else
    \begingroup
      \edef\exfs@tempb{\csname\exfs@base@family @orn\@roman#1\endcsname}%
      \expandafter\expandafter\expandafter\exfs@split@orndef
        \expandafter\string\exfs@tempb\@nil
      \selectfont\char\exfs@tempa
    \endgroup
  \fi}
```

\nfssextcfr@MT@Hook \Microtype@Hook BEGIN add microtype hooks Partly from microtype docs; partly from MinionPro package

```
\def\nfssextcfr@MT@Hook{%
  \DeclareMicrotypeVariants*{2,2d,2dj,2j,dj,e,h,l}% is this necessary or would the previous line be enough?
}
\@ifpackageloaded{microtype}{%
  \PackageWarning{nfssext-cfr}{%
    You have loaded me (or a font support package which loads me)\MessageBreak
    after loading microtype, but microtype should be loaded after\MessageBreak
    all font defaults have been setup}%
  \nfssextcfr@MT@Hook
}{%
  \@ifundefined{Microtype@Hook}{%
    \let\Microtype@Hook\nfssextcfr@MT@Hook}{% MinionPro has \global before this
    \g@addto@macro\Microtype@Hook{\nfssextcfr@MT@Hook}}}
```

END

6.2 NNFSS

This code was written for the *current* (New) New Font Selection Scheme (2020–). It should not generally be loaded on older kernels.

`nfssext-cfr-nnfss` (*pkg.*)

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{svn-prov}
\ProvidesPackageSVN[\filebase-nnfss.sty]{$Id: nfssext-cfr.dtx 10366 2024-09-18 14:25:21Z cfrees $}[v1.0 \revinfo{} extended New New Font Selection Scheme (NNFSS) based on 2003/03/14 v1.2 Experimental NFSS Extensions]
\DefineFileInfoSVN
```

END added

`\exfs@tempa`
`\exfs@tempb`
`\exfs@tempf`

Scratch variables.

```
\newcommand*{\exfs@tempa}{}
\newcommand*{\exfs@tempb}{}
```

:end-added BEGIN added (cfr): extra variable (`\exfs@tempf`)

```
\newcommand*{\exfs@tempf}{}
```

We want to track cases where missing fonts get defined into existence. To do this we define an additional macro each time `\wrong@fontshape` is called. This is based on two proposals by Max Chernoff, but the implementation is different.

```
\ExplSyntaxOn
\hook_gput_code:nnn { cmd/wrong@fontshape/before } { . }
{
  \global\expandafter\expandafter\expandafter\let
  \expandafter \csname exfs@fake@ \curr@fontshape\endcsname\relax
}
\ExplSyntaxOff
```

`\exfs@info` Custom logging

```
\newcommand \exfs@info[2][nfssext-cfr]{%
  \ifexfs@debug
    \PackageWarning{#1}{Info: #2}%
  \else
    \PackageInfo{#1}{#2}%
  \fi
}
```

END added

`\exfs@normalise` BEGIN added (cfr): normalise

```
\newcommand* \exfs@normalise[1]{%
  \ifcsname exfs@fake@\curr@fontshape\endcsname
    \exfs@info{Current font '\curr@fontshape' is fake.\MessageBreak
      Normalising}%
    \expandafter\csname f@#1\endcsname{\csname #1default\endcsname}\selectfont
    \ifcsname exfs@fake@\curr@fontshape\endcsname
      \f@series{\seriesdefault}\f@shape{\shapedefault}\selectfont
      \ifcsname exfs@fake@\curr@fontshape\endcsname
```

This might happen, I think, if we're in a swash family or specialist encoding where the default series and shape aren't available. All bets are off here so try to bale out as gracefully as we can.

```
        \normalfont
        \ifcsname exfs@fake@\curr@fontshape\endcsname
```

This definitely oughtn't to happen, though - things are really screwed up at this point - so error.

```
          \PackageError{nfssext-cfr}{Default font appears to be fake!\MessageBreak
            Switch \textbackslash normalfont yielded '\curr@fontshape'}
          {This is highly unlikely, so the bug is probably in the phenomena\MessageBreak
            rather than the noumena}%
        \fi
      \fi
    \fi
    \exfs@info{Normalised to '\curr@fontshape'}%
  \else
    \exfs@info{Current font '\curr@fontshape' appears real}%
  \fi
}
```

END added

`\exfs@try@family` Modified from nfssext? Or modified from older nfssext-cfr?

```
\newcommand*{\exfs@try@family}[2][]{%
  \let\exfs@tempa\relax
```

END added

trans: group is requisite here else LaTeX thinks the family real regardless

```
  \begingroup % fel arall, bydd latex yn credu bod y family yn go iawn beth bynnag
```

(o leiaf bydd latex yn dweud felly)

```
    \exfs@info{Trying Font family '\f@encoding/#2'}%
    \fontfamily{#2}\try@load@fontshape
```

`\curr@fontshape` holds the target shape - not the current one - after an unsuccessful attempt to load **family** with `\try@load@fontshape`. This won't work for series/shape as `\curr@fontshape` holds the current one in that case

```
    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
      \edef\exfs@tempa{#1}%
      \ifx\exfs@tempa\@empty
        \PackageWarning{nfssext}{%
          Font family '\f@encoding/#2' not available\MessageBreak
          Ignoring font switch}%
      \else
        \exfs@info[nfssext]{%
          Font family '\f@encoding/#2' not available\MessageBreak
          Font family '\f@encoding/#1' tried instead}%
        \exfs@try@family{#1}%
      \fi
    \else
      \exfs@info{Loading font family '\f@encoding/#2'}%
      \gdef\exfs@tempa{\fontfamily{#2}\selectfont}%
    \fi
  \endgroup
  \exfs@tempa}
```

`\exfs@try@series` BEGIN added (cfr)

```
\newcommand*{\exfs@try@series}[2][]{%
```

We don't hand instructions to the kernel unless we know they'll succeed b/c the results are too unpredictable under NNFSS.

Changing directly only produces usable results for series defined in the 'table' of font changes. But using higher level kernel interfaces for tests doesn't work because spurious fonts get defined, which only seem to exist. Theoretically, we might as well use the existing kernel's macros since we're already damned by reliance on internals anyway. But then everything needs disentangling. So it's easier to just adapt previous tests, even though it partially duplicates what the kernel does. (But it isn't the mess swash is . . .).

```
  \let\exfs@targetseries\relax
  \edef\exfs@tempa{#2}%
  \ifx\f@series\exfs@tempa\relax
  \exfs@info{Current (\f@series) matches target (#2) series.\MessageBreak
    Ignoring font switch}%
  \else
    \begingroup
      \exfs@normalise{series}%
```

```
   \edef\exfs@tempa{\f@encoding/\f@family/#2/\f@shape}%
   \ifcsname \exfs@tempa\endcsname
    \exfs@info{Switching series: \f@series\space -> #2}%
    \gdef\exfs@targetseries{\fontseries{#2}\selectfont}%
   \else
    \edef\exfs@reserved{#1}%
    \ifx\exfs@reserved\@empty
     \PackageWarning{nfssext-cfr}{%
      Font series '\f@encoding/\f@family/#2/\f@shape' not available\MessageBreak
      Ignoring font change}%
    \else
     \PackageWarning{nfssext-cfr}{%
      Font series '\f@encoding/\f@family/#2/\f@shape' not available\MessageBreak
      Trying '\f@encoding/\f@family/#1/\f@shape'}%
      \exfs@try@series{#1}%
    \fi
   \fi
  \endgroup
  \exfs@targetseries
 \fi}
```

`\exfs@try@shapeshift` Attempt to leverage kernel's mechanism.

```
\def\exfs@try@shapeshift#1{%
 \edef\exfs@targetshape{\csname #1default\endcsname}%
 \ifx\f@shape\exfs@targetshape\relax
  \exfs@info{Current (\f@shape) matches target (#1) shape.\MessageBreak
   Ignoring font switch}%
 \else
  \not@math@alphabet\edef\exfs@targetshape\relax
  \exfs@info{\f@shape\space -> \exfs@targetshape\MessageBreak
   Trying \f@encoding/\f@family/\f@series/\exfs@targetshape}%
```

We *do* want the kernel's substitution mechanism here?

```
  \fontshape{\exfs@targetshape}\selectfont
 \fi}
```

`\exfs@swshape` Switching to swash is now far more complicated with (I presume) the attendant overhead, but the kernel's approach just won't work here[9]. This will become `\swshape` if `compat` isn't enabled.

```
\newcommand* \exfs@swshape{%
 \let\exfs@targetsw\relax
 \begingroup % angen neu beidio? angen - bendant!
```

Try kernel or configured default first so we get swash from current family etc. if available

[9] Implementing swash as a *shape* isn't workable for fonts I've packaged, so I've made no attempt to follow the kernel here as I do for small-caps italic etc. We're back to the single axis/multiple aspect problem which NFSS created by ignoring small-caps/italic and width/weight combinations. It may, in fact, be wrong-headed to follow the kernel *at all* here. Perhaps it would be better to just provide the original implementation and some compatibility option for people who also need swash shapes in the same document?

```
    \edef\exfs@tempa{\f@encoding/\f@family/\f@series/\swdefault}%
    \edef\exfs@tempa@fake{exfs@fake@\exfs@tempa}%
    \ifcsname \exfs@tempa\endcsname
      \ifcsname exfs@fake@\exfs@tempa\endcsname
        \exfs@swfamily
      \else
        \gdef\exfs@targetsw{\fontshape{\swdefault}\selectfont}% kernel
      \fi
    \else
      \exfs@swfamily
    \fi
  \endgroup
  \ifx\exfs@targetsw\relax
    \PackageWarning{nfssext-cfr}{%
      Cannot find any route to swash.\MessageBreak
      Are you sure one is available?}%
  \else
    \exfs@targetsw
    \exfs@info{Switch to swash resulted in '\curr@fontshape'}%
  \fi
}
```

`\exfs@swfamily` This is the guts of `\exfs@swshape`.

```
\newcommand* \exfs@swfamily{%
  \let\exfs@targetsw\relax
  \begingroup
```

Try nfssext-cfr family switch & our default or configured

```
    \let\exfs@tempa\f@family
    \exfs@merge@families{w}%
    \ifx\exfs@tempa\f@family % try merge with current shape
```

Try switching to upright etc. first

```
      \fontshape{n}%
      \exfs@merge@families{w}% up & merge
      \if\exfs@tempa\f@family
```

Try switching to `\swshapedefault` first

```
        \fontshape\swshapedefault
        \exfs@merge@families{w}% up & nfssext-cfr default/configured
        \if\exfs@tempa\f@family
```

Use nfssext family switch & default or configured

```
          \exf@try@family{\expandafter\exfs@get@base\f@family\@nil w}% nfssext switch
          \if\exfs@tempa\f@family % nfssext switch
            \fontshape\swshapedefault\exfs@try@family{%
            \expandafter\exfs@get@base\f@family\@nil w}% nfssext switch & shape
            \if\exfs@tempa\f@family
              \relax % rhodd y ffidl yn y to (give up)
            \else
              \gdef\exfs@targetsw{%
                \fontshape\swshapedefault\expandafter\fontfamily{%
                \exfs@get@base\f@family\@nil w}\selectfont
              }% nfssext switch & shape
            \fi % nfssext switch & shape
          \else
            \gdef\exfs@targetsw{\expandafter\fontfamily{%
              \exfs@get@base\f@family\@nil w}\selectfont
            }% nfssext switch
          \fi % nfssext switch
        \else
          \gdef\exfs@targetsw{%
            \fontshape\swshapedefault\exfs@merge@families{w}%
          }%  up & nfssext-cfr default/configured
        \fi % up & nfssext-cfr default/configured
      \else
        \gdef\exfs@targetsw{%
          \fontshape{n}\exfs@merge@families{w}%
        }% up & merge
      \fi % up & merge
    \else
      \gdef\exfs@targetsw{\exfs@merge@families{w}}% merge with current shape
    \fi % merge with current shape
  \endgroup
}
```

END added

\exfs@get@base
\exfs@get@variants
\exfs@next
\exfs@shift
\exfs@first
\exfs@part
\exfs@second

Utlities

```
\def\exfs@get@base#1#2#3#4\@nil{#1#2#3}
```

BEGIN added (cfr): more `\exfs@` commands (`get@variants`, `next`, `shift`, `first`, `part`, `second`)

```
\def\exfs@get@variants#1#2#3#4\@nil{#4}
\def\exfs@next#1#2\@nil{#1}
\def\exfs@shift#1#2\@nil{#2}
\def\exfs@first#1#2\@nil{#1}
\def\exfs@part#1#2\@nil{#2}
\def\exfs@second#1#2#3\@nil{#2}
```

\exfs@series@splitter

Common method for dealing with weight and width.

```
\def\exfs@series@splitter#1{%
  \edef\exfs@weight{\expandafter\exfs@first#1\@nil}%
  \edef\exfs@width{\expandafter\exfs@shift#1\@nil}%
```

Two char width only or two char weight

```
  \if\exfs@weight u\exfs@check@cx{u}%
    \else\if\exfs@weight e\exfs@check@cx{e}%
      \else\if\exfs@weight s\exfs@check@cx{s}%
        \else\if\exfs@weight d\exfs@check@cx{d}%
```

```
          \else\ifx\exfs@width\@empty % m dealt with elsewhere
```

Single character width

```
            \if\exfs@weight c\def\exfs@width{c}\let\exfs@weight\@empty
            \else\if\exfs@weight x\def\exfs@width{x}\let\exfs@weight\@empty
          \fi\fi
  \fi\fi\fi\fi\fi
  \exfs@info{#1 -> \exfs@weight:\exfs@width;}%
}
```

\exfs@check@cx

Auxiliary for \exfs@check@cx

```
\def\exfs@check@cx#1{%
  \edef\exfs@tempa{\expandafter\exfs@first\exfs@width\@nil}%
  \if\exfs@tempa c\edef\exfs@width{#1c}\let\exfs@weight\@empty
    \else\if\exfs@tempa x\edef\exfs@width{#1x}\let\exfs@weight\@empty
    \else\edef\exfs@weighta{%
        \exfs@weight\exfs@tempa
      }\let\exfs@weight\exfs@weighta
      \edef\exfs@widtha{%
        \expandafter\exfs@shift\exfs@width\@nil
      }\let\exfs@width\exfs@widtha
  \fi\fi
}
```

END added

\lnstyle
\osstyle
\infstyle
\instyle
\sustyle
\swstyle

Unmodified from nfssext. Anniffiniedig -> undefined in the kernel.

```
\DeclareRobustCommand{\lnstyle}{%% anniffiniedig
  \not@math@alphabet\lnstyle\relax
  \exfs@try@family[\expandafter\exfs@get@base\f@family\@nil]%
                  {\expandafter\exfs@get@base\f@family\@nil x}}
\DeclareRobustCommand{\osstyle}{%% anniffiniedig
  \not@math@alphabet\osstyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil j}}
\DeclareRobustCommand{\infstyle}{%% anniffiniedig
  \not@math@alphabet\infstyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil 0}}
\let\instyle\infstyle
\DeclareRobustCommand{\sustyle}{%% anniffiniedig
  \not@math@alphabet\sustyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil 1}}
\DeclareRobustCommand{\swstyle}{%% anniffiniedig
  \not@math@alphabet\swstyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil w}}
```

BEGIN added (cfr) - merge families.

NNFSS (unsurprisingly) does nothing here, so this is unproblematic. The following depends **absolutely** on **complete** adherence to berry names.

\ifexfs@added
\exfs@merge@families

\exfs@merge@families is used in the macros recommended for switching the style of figures, activating swash and other variants etc.

```
\newif\ifexfs@added
\newcommand*\exfs@merge@families[1]{%
 \edef\exfs@vartomerge{#1}%
 \edef\exfs@variants{\expandafter\exfs@get@variants\f@family\@nil}%
 \exfs@info{Trying to merge variants #1 and \exfs@variants}%
 \edef\tempo{2j}%
 \let\exfs@tempq\@empty
 \def\exfs@tempg{}%
 \exfs@addedfalse
```

Check whether there are variants - if not just use the requested addition.

```
 \ifx\exfs@variants\@empty
  \edef\exfs@tempq{\exfs@vartomerge}%
  \exfs@addedtrue
 \else
  \gdef\set{0,1,2,a,d,e,f,h,j,l,p,q,s,t,v,w}% these are the variants to consider - the order here and in the font name is crucial
  \ifx\tempo\exfs@vartomerge
   \@for \xx:=\set \do {%
```

Check whether there are variants left - if not set the 'next variant' to empty

```
    \ifx\exfs@variants\@empty
     \let\exfs@nextvariant\@empty
    \else
```

O/w get the next variant

```
     \edef\exfs@nextvariant{\expandafter\exfs@next\exfs@variants\@nil}%
    \fi
```

If the next variant is 2 or j, ignore it

```
    \if\exfs@nextvariant 2%
     \edef\exfs@variants{\expandafter\exfs@shift\exfs@variants\@nil}%
    \fi
    \if\exfs@nextvariant j% if the next variant is j, ignore it
     \edef\exfs@variants{\expandafter\exfs@shift\exfs@variants\@nil}%
    \fi
```

See if the current value is either 2 or j and add it if so and if needed

```
    \if\xx 2%
     \edef\exfs@tempg{\exfs@tempg\xx}%
    \else
     \if\xx j% if the current value is j, we're done
      \edef\exfs@tempq{\exfs@tempg\xx\exfs@variants}%
      \let\exfs@variants\@empty
      \exfs@addedtrue
     \else
```

o/w see if the current value matches the next variant

```
      \ifx\xx\exfs@nextvariant
       \edef\exfs@tempg{\exfs@tempg\xx}%
       \edef\exfs@variants{\expandafter\exfs@shift\exfs@variants\@nil}%
```

```
            \fi
          \fi
        \fi
      }%
    \else
      \@for \xx:=\set \do {%
```

Check whether there are variants left and, if not, add the addition if needed

```
        \ifx\exfs@variants\@empty
          \ifexfs@added
          \else
            \edef\exfs@tempq{\exfs@tempg\exfs@vartomerge}%
            \exfs@addedtrue
          \fi
        \else
```

o/w get the next variant

```
          \edef\exfs@nextvariant{\expandafter\exfs@next\exfs@variants\@nil}%
```

If the new token equals the next variant, combine whatever is saved in `\exfs@tempg` with whatever remains in `\exfs@variants`

```
          \ifx\exfs@nextvariant\exfs@vartomerge
            \edef\exfs@tempq{\exfs@tempg\exfs@variants}%
            \exfs@addedtrue
            \let\exfs@variants\@empty
          \else
```

o/w, if the current value matches the requested addition, add it in

```
            \ifx\exfs@vartomerge\xx
              \edef\exfs@tempq{\exfs@tempg\xx\exfs@variants}%
              \exfs@addedtrue
              \let\exfs@variants\@empty
            \else
```

o/w, if the current value matches the next variant, shift

```
              \ifx\exfs@nextvariant\xx
                \edef\exfs@tempg{\exfs@tempg\xx}%
                \edef\exfs@variants{\expandafter\exfs@shift\exfs@variants\@nil}%
              \fi
            \fi
          \fi
        \fi
      }%
    \fi
  \fi
  \ifx\exfs@tempq\@empty
    \PackageError{nfssext-cfr}{Something is wrong here. Ignoring font switching command.}{}%
  \else
    \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil \exfs@tempq}%
  \fi
}
```

\pstyle \ostyle \postyle

Commands for switching to proportional and/or oldstyle figures. Compare \ostyle with nfssext's \osstyle above. These macros (and the block which follows) all require merging Berry names but not unmerging.

```
\DeclareRobustCommand{\pstyle}{%% anniffiniedig  proportional figures
  \not@math@alphabet\pstyle\relax
  \exfs@merge@families{2}}
\DeclareRobustCommand{\ostyle}{%% anniffiniedig  oldstyle figures (cf. original osstyle above)
  \not@math@alphabet\ostyle\relax
  \exfs@merge@families{j}}
```

Combined command for proportional oldstyle

```
\DeclareRobustCommand{\postyle}{%% anniffiniedig
  \not@math@alphabet\postyle\relax
  \exfs@merge@families{2j}}
```

\tistyle \ltstyle \ofstyle \altstyle \regstyle \embossstyle \ornamentalstyle \swashstyle \shstyle \qtstyle

These macros again require merging, but not unmerging, names.

```
\DeclareRobustCommand{\tistyle}{%% anniffiniedig titling/display
  \not@math@alphabet\tistyle\relax
  \exfs@merge@families{d}}
```

Note that this command is for use when the light version is a separate family rather than a weight variant (e.g. when you've got light, light bold etc. as well as regular weights)

```
\DeclareRobustCommand{\ltstyle}{%% anniffiniedig
  \not@math@alphabet\ltstyle\relax
  \exfs@merge@families{l}}
```

Let's hope there aren't any fonts with a light family *and* an outline/openface/blank version

```
\DeclareRobustCommand{\ofstyle}{%% anniffiniedig
  \not@math@alphabet\ofstyle\relax
  \exfs@merge@families{l}}
\DeclareRobustCommand{\altstyle}{%% anniffiniedig alternative style
  \not@math@alphabet\altstyle\relax
  \exfs@merge@families{a}}
\DeclareRobustCommand{\regstyle}{%% anniffiniedig 'regular' style
  \not@math@alphabet\regstyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil}}
\DeclareRobustCommand{\embossstyle}{%% anniffiniedig
  \not@math@alphabet\embossstyle\relax
  \exfs@merge@families{e}}
\DeclareRobustCommand{\ornamentalstyle}{%% anniffiniedig intended primarily for decorative initial fonts etc.
  \not@math@alphabet\ornamentalstyle\relax
  \exfs@merge@families{p}}
\DeclareRobustCommand{\qtstyle}{%% anniffiniedig quotation style (assumes sans)
  \not@math@alphabet\qtstyle\relax
  \sffamily
  \exfs@merge@families{q}}
```

```
\DeclareRobustCommand{\shstyle}{%% anniffiniedig
  \not@math@alphabet\shstyle\relax
  \exfs@merge@families{h}}
\DeclareRobustCommand{\swashstyle}{%% anniffiniedig   an attempt to improve on \swstyle
  \not@math@alphabet\swashstyle\relax
  \exfs@merge@families{w}}
```

\tmstyle
\tvstyle

Macros to switch between monowidth and variable typewriter. These need to unmerge before merging. We need to unmerge sans as well as the other kind of typewriter.

```
\DeclareRobustCommand{\tmstyle}{%% anniffiniedig       monowidth typewriter
  \not@math@alphabet\tmstyle\relax
  \exfs@unmerge@families{s}%
  \exfs@unmerge@families{v}%
  \exfs@merge@families{t}}
\DeclareRobustCommand{\tvstyle}{%% anniffiniedig       variable width typewriter
  \not@math@alphabet\tvstyle\relax
  \exfs@unmerge@families{s}%
  \exfs@unmerge@families{t}%
  \exfs@merge@families{v}}
```

BEGIN added (cfr) - unmerge families

\ifexfs@take
\exfs@unmergefamilies

Define the macro needed to do the unmerges.

```
\newif\ifexfs@take
\newcommand*\exfs@unmerge@families[1]{%
  \edef\exfs@tempf{#1}%
  \edef\tempa{\expandafter\exfs@get@variants\f@family\@nil}%
  \let\exfs@tempq\@empty
  \edef\exfs@tempg{}%
  \exfs@taketrue
```

Check whether there are variants - if not do nothing

```
  \ifx\tempa\@empty
    \edef\exfs@tempq{}%
  \else
```

o/w go through the variants to find the one to delete

```
    \loop
```

Get the next variant

```
      \edef\exfs@tempn{\expandafter\exfs@next\tempa\@nil}%
```

See if the next variant is the thing we seek and, if so, eliminate it

```
      \ifx\exfs@tempf\exfs@tempn
        \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
        \edef\exfs@tempq{\exfs@tempg\tempa}%
        \exfs@takefalse
```

o/w save the next variant and move on if any variants remain

```
      \else
        \edef\exfs@tempg{\exfs@tempg\exfs@tempn}%
        \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
        \ifx\tempa\@empty% if there are no variants left, we're done
          \edef\exfs@tempq{\exfs@tempg}%
          \exfs@takefalse
        \fi
      \fi
      \ifexfs@take % \fi yn \repeat
    \repeat
  \fi
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil \exfs@tempq}%
}
```

`\tstyle` `\lstyle` These are simple unmerges, with no merging necessary.

```
\DeclareRobustCommand{\tstyle}{%% anniffiniedig  tabular figures
  \not@math@alphabet\tstyle\relax
  \exfs@unmerge@families{2}}
\DeclareRobustCommand{\lstyle}{%% anniffiniedig  lining figures (cf. command above)
  \not@math@alphabet\lstyle\relax
  \exfs@unmerge@families{j}}
```

`\tlstyle` `\plstyle` `\tostyle` Simple combinations for combined figure styles.

Make a combined command for tabular lining

```
\DeclareRobustCommand{\tlstyle}{%% anniffiniedig
  \lstyle\tstyle}
```

Proportional lining

```
\DeclareRobustCommand{\plstyle}{%% anniffiniedig
  \lstyle\pstyle}
```

Tabular oldstyle ?!

```
\DeclareRobustCommand{\tostyle}{%% anniffiniedig
  \ostyle\tstyle}
```

END added

`\sidefault` `\sishape` `si` is italic sc[10]. We use the original definition for the default and then set up rules for font shape changes which try `scit` and `scsl` before falling back to `si`.

```
\newcommand*{\sidefault}{si}% anniffiniedig
```

Well i brofi `si` & yna `scit` ond wn i ddim sut i wneud hon gyda'r stwff newydd . . . . I was going to deprecate the `si` macros, but the truth is they are much nicer than having to combine macros for those cases when you really do want precisely italic small-caps. They are also much more robust than somebody trying to force

[10] That is, italic small-caps *was* `si`. These days, things are more complicated . . . .

things with `\fontshape{⟨si⟩}\selectfont`, so, on reflection, it seems better to retain the interface, even if the implementation isn't as straightforward as I'd like.

```
\DeclareRobustCommand{\sishape}{%
  \exfs@try@shapeshift{si}}
```

Kernel virtuals: ulc upper/lower case up upright. nfssext virtuals? or nfssext-cfr virtuals? or? It would be better to try `si`, `scit` and `scsl`, but that doesn't seem possible

```
\DeclareFontShapeChangeRule {n}{si}{scit}{scsl}% current; request (& trydedd); dewisiad cyntaf; ail ddewisiad
\DeclareFontShapeChangeRule {it}{si}{scit}{scsl}
\DeclareFontShapeChangeRule {sl}{si}{scit}{scsl}
\DeclareFontShapeChangeRule {sc}{si}{scit}{scsl}
```

Current shape is si => font support hasn't been updated

```
\DeclareFontShapeChangeRule {si}{sc} {si} {}
\DeclareFontShapeChangeRule {si}{it} {si} {}
\DeclareFontShapeChangeRule {si}{sl} {scsl} {si}
\DeclareFontShapeChangeRule {si}{scit} {scit} {si}% rhag ofn?!
\DeclareFontShapeChangeRule {si}{ulc} {it} {}
\DeclareFontShapeChangeRule {si}{up}{sc}{}
```

```
\ExplSyntaxOn
```

The need for overwriting can be avoided by changing the `.fd` files, but `compat` supports packages I don't know about[11] . . .

```
\bool_if:NT \g__exfs_compat_bool
{
  \DeclareFontShapeChangeRule {it}{sc}{si}{scsl}
  \DeclareFontShapeChangeRule{sl}{sc}{scsl}{si}
```

Gofyn am italic etc. | Ask about italic etc. Sylwadau tebygol yma . . . | Like comments here . . .

```
  \DeclareFontShapeChangeRule {sc}{it} {si} {scsl}
  \DeclareFontShapeChangeRule {sc}{sl} {scsl} {si}
  \DeclareFontShapeChangeRule {scsl}{it} {si} {scsl}
}
\ExplSyntaxOff
```

```
\DeclareFontShapeChangeRule {ui}{sc}{scit}{scsl}
\DeclareFontShapeChangeRule {ui}{scsl}{scsl}{scit}
\DeclareFontShapeChangeRule {ui}{it}{it}{ui}
\DeclareFontShapeChangeRule {ui}{ri}{ri}{ui}
\DeclareFontShapeChangeRule {ui}{up}{n}{}
\DeclareFontShapeChangeRule {ui}{ulc}{ui}{}
```

```
\DeclareFontShapeChangeRule {ri}{sc}{scit}{scsl}
\DeclareFontShapeChangeRule {ri}{scsl}{scsl}{scit}
\DeclareFontShapeChangeRule {ri}{it}{it}{ri}
\DeclareFontShapeChangeRule {ri}{ui}{ui}{ri}
```

[11] Or haven't published? I haven't actually tried it with those.

Kernel virtuals: `ulc` upper/lower case up upright

```
\DeclareFontShapeChangeRule {ri}{up}{n}{}
\DeclareFontShapeChangeRule {ri}{ulc}{ri}{}
```

nfssext virtuals? or nfssext-cfr virtuals? or?

```
\DeclareFontShapeChangeRule {ol} {sc} {scol} {} % <- seiliedig ar nfssext-cfr-nfss.sty \scshape
\DeclareFontShapeChangeRule {ol} {ulc} {ol} {}
\DeclareFontShapeChangeRule {ol} {up} {ol} {}

\DeclareFontShapeChangeRule {scol} {sc} {scol} {}
\DeclareFontShapeChangeRule {scol} {ulc} {ol} {}
\DeclareFontShapeChangeRule {scol} {up} {scol} {}

\DeclareFontShapeChangeRule {u}{sc}{su}{} % <- seiliedig ar nfssext-cfr-nfss.sty \scshape
\DeclareFontShapeChangeRule {su}{ulc}{u}{}
\DeclareFontShapeChangeRule {sc}{u}{su}{}
\DeclareFontShapeChangeRule {su}{u}{su}{}
\DeclareFontShapeChangeRule {su}{sc}{su}{}
```

BEGIN added (cfr)

cfr: is this how outline shapes should be handled?

`\oldefault` `\olshape` `\scoldefault` `\scolshape`

Outline

```
\newcommand*{\oldefault}{ol}% anniffiniedig
\DeclareRobustCommand{\olshape}{%% anniffiniedig
  \exfs@try@shapeshift{ol}}
\newcommand*{\scoldefault}{scol}% anniffiniedig
\DeclareRobustCommand{\scolshape}{%% anniffiniedig
  \exfs@try@shapeshift{scol}}
```

`\udefault` `\ushape` `\scudefault` `\scushape`

Underlined?? Fudge <- ??

```
\newcommand*{\udefault}{u}% anniffiniedig
\DeclareRobustCommand{\ushape}{%% anniffiniedig
  \exfs@try@shapeshift{u}}
\newcommand*{\scudefault}{su}% anniffiniedig
\DeclareRobustCommand{\scushape}{%% anniffiniedig
  \exfs@try@shapeshift{scu}}
```

`\uidefault` `\uishape` `\ridefault` `\rishape`

Upright and reverse italic

```
\newcommand*{\uidefault}{ui}% anniffiniedig
\DeclareRobustCommand{\uishape}{% anniffiniedig
  \exfs@try@shapeshift{ui}}
```

Can i do this for reverse italic?

```
\newcommand*{\ridefault}{ri}% anniffiniedig
\DeclareRobustCommand{\rishape}{% anniffiniedig
  \exfs@try@shapeshift{ri}}
```

END added BEGIN added (cfr) - merge width changes into series

\exfs@merge@width Previously dependent on incorrect series names.

```
\newcommand*{\exfs@merge@width}[1]{%
```

cfr-added

Dibynnodd y côd gwreiddiol ar *mc* etc. & r'odd hynny'n anghywir

Instead of merging or unmerging cyclically, which means keeping track of everything, we split the current series (which requires some juggling, but hopefully less) and use the results.

```
    \exfs@series@splitter{\f@series}%
    \edef\exfs@temph{#1}%
    \if\exfs@temph m\ifx\exfs@weight\@empty\else\let\exfs@temph\@empty\fi\fi
    \if\exfs@weight m\ifx\exfs@temph\@empty\else\let\exfs@weight\@empty\fi\fi
    \edef\exfs@series{\exfs@weight\exfs@temph}%
```

end cfr-added

```
    \exfs@info{Trying \exfs@series}%
 \exfs@try@series{\exfs@series}}
```

\regwidth 'Regular' width requires conditionally adding 'm'.

```
\DeclareRobustCommand{\regwidth}{%% anniffiniedig
 \not@math@alphabet\regwidth\relax
```

cfr altered

```
    \exfs@merge@width{m}}
```

\nwdefault
\nwwidth
\cddefault
\cdwidth
\ecdefault
\ecwidth
\ucdefault
\ucwidth

Condensed widths.

```
\newcommand*{\nwdefault}{c}% anniffiniedig
\DeclareRobustCommand{\nwwidth}{%% anniffiniedig ond rheolau
 \not@math@alphabet\nwwidth\relax
 \exfs@merge@width{\nwdefault}}% neu \exfs@try@series ?
\newcommand*{\cddefault}{c}% anniffiniedig
\DeclareRobustCommand{\cdwidth}{% anniffiniedig ond rheolau
 \not@math@alphabet\cdwidth\relax
 \exfs@merge@width{\cddefault}}% neu \exfs@try@series ?
\newcommand*{\ecdefault}{ec}% anniffiniedig
\DeclareRobustCommand{\ecwidth}{%% anniffiniedig ond rheolau
 \not@math@alphabet\ecwidth\relax
 \exfs@merge@width{\ecdefault}}% neu \exfs@try@series ?
\newcommand*{\ucdefault}{uc}% anniffiniedig
\DeclareRobustCommand{\ucwidth}{%% anniffiniedig
 \not@math@alphabet\ucwidth\relax
 \exfs@merge@width{\ucdefault}}
```

\etdefault
\etwidth
\epdefault
\epwidth
\exdefault
\exwidth
\uxdefault
\uxwidth

Extended/expanded widths.

```
\newcommand*{\etdefault}{x}% anniffiniedig
```

```
\DeclareRobustCommand{\etwidth}{%% anniffiniedig
 \not@math@alphabet\etwidth\relax
 \exfs@merge@width{\etdefault}}% neu \exfs@try@series ?
\newcommand*{\epdefault}{x}% anniffiniedig
\DeclareRobustCommand{\epwidth}{%% anniffiniedig ond rheolau
 \not@math@alphabet\epwidth\relax
 \exfs@merge@width{\epdefault}}% neu \exfs@try@series ?
\newcommand*{\exdefault}{ex}% anniffiniedig
\DeclareRobustCommand{\exwidth}{%% anniffiniedig
 \not@math@alphabet\exwidth\relax
 \exfs@merge@width{\exdefault}}
\newcommand*{\uxdefault}{ux}% anniffiniedig
\DeclareRobustCommand{\uxwidth}{%% anniffiniedig
 \not@math@alphabet\uxwidth\relax
 \exfs@merge@width{\uxdefault}}
```

\mdwdefault \mdwidth Medium.

```
\newcommand*\mdwdefault{m}
\DeclareRobustCommand{\mdwidth}{%% anniffiniedig
  \not@math@alphabet\mdwidth\relax
  \exfs@merge@width{\mdwdefault}}
```

Posibl ond bydda i'n colli achosion yn siwr. | Possible but I'd lose cases for sure. Hefyd hoffwn i ddim dyfalu pa rheolau y bydden nhw eu dewis. | Also I wouldn't like to guess which rules they'll choose.

BEGIN added (cfr) merge weight changes into series

\exfs@merge@weight The pay off for setting up series splitting is that we can reuse the method here and, as in the case of width, the definition is greatly simplified[12].

```
\newcommand*{\exfs@merge@weight}[1]{%
  \exfs@series@splitter{\f@series}%
```

Save current series so we can test for change

```
  \let\exfs@tempg\f@series
  \def\exfs@temph{#1}%
  \if\exfs@temph m\relax
    \ifx\exfs@width\@empty\relax
    \else
      \let\exfs@temph\@empty
    \fi
  \fi
  \edef\exfs@series{\exfs@temph\exfs@width}%
  \ifx\exfs@temph\exfs@series
    \exfs@info{Trying \exfs@series}%
    \exfs@try@series{\exfs@series}%
  \else
    \exfs@info{Trying \exfs@series, favouring \exfs@weight}%
    \exfs@try@series[\exfs@weight]{\exfs@series}% assume user wants to change weight even if this changes back to the default width
  \fi}
```

[12] At least if you don't look at the splitter code.

`\mbdefault` `\mbweight` `\bddefault` `\bfweight` `\bdweight`

Ref.: `sources2e.pdf` and/or stripped code in `base`.

The annotation ‘anniffiniedig’ indicates the macro is *not* defined by the kernel as of 2024. The addition ‘ond rheolau’ means there are nonetheless relevant rules. The comment ‘dim byd i’w gael eu wneud yma’ indicates that defining the default is now sufficient and no additional font switch or text command is required.

```
\newcommand*{\mbdefault}{sb}% dim byd i'w gael ei wneud yma
\DeclareRobustCommand{\mbweight}{%% anniffiniedig
  \not@math@alphabet\mbweight\relax
  \exfs@merge@weight{\mbdefault}}
\newcommand*{\bddefault}{b}% dim byd i'w gael ei wneud yma
\DeclareRobustCommand{\bfweight}{%% anniffiniedig
  \not@math@alphabet\bfweight\relax
  \exfs@merge@weight{\bddefault}}
\DeclareRobustCommand{\bdweight}{%% anniffiniedig
  \not@math@alphabet\bdweight\relax
  \exfs@merge@weight{\bddefault}}
```

`\mwdefault` `\mdweight`

Regular, medium, default are all irregular, exceptional, fraught.

```
\newcommand*\mwdefault{m}
\DeclareRobustCommand{\mdweight}{%% anniffiniedig
  \not@math@alphabet\mdweight\relax
  \exfs@merge@weight{\mwdefault}}
```

`\dbdefault` `\dbweight` `\sbdefault` `\sbweight` `\ebdefault` `\ebweight` `\ubdefault` `\ubweight`

Heavy weights.

```
\newcommand*{\dbdefault}{db}% anniffiniedig
\DeclareRobustCommand{\dbweight}{%% anniffiniedig
 \not@math@alphabet\dbweight\relax
 \exfs@merge@weight{\dbdefault}}
\newcommand*{\sbdefault}{sb}% anniffiniedig
\DeclareRobustCommand{\sbweight}{%% anniffiniedig ond rheolau
 \not@math@alphabet\sbweight\relax
 \exfs@merge@weight{\sbdefault}}% neu \exfs@try@series ?
\newcommand*{\ebdefault}{eb}% anniffiniedig
\DeclareRobustCommand{\ebweight}{%% anniffiniedig ond rheolau
 \not@math@alphabet\ebweight\relax
 \exfs@merge@weight{\ebdefault}}% neu \exfs@try@series ?
\newcommand*{\ubdefault}{ub}% anniffiniedig
\DeclareRobustCommand{\ubweight}{%% anniffiniedig ond rheolau
 \not@math@alphabet\ubweight\relax
 \exfs@merge@weight{\ubdefault}}% neu \exfs@try@series ?
```

`\lgdefault` `\lgweight` `\eldefault` `\elweight` `\uldefault` `\ulweight`

Light weights.

```
\newcommand*{\lgdefault}{l}% anniffiniedig
```

Note - use this if light is a variant weight, rather than a separate family

```
\DeclareRobustCommand{\lgweight}{%% anniffiniedig ond rheolau
 \not@math@alphabet\lgweight\relax
 \exfs@merge@weight{\lgdefault}}% neu \exfs@try@series ?
\newcommand*{\eldefault}{el}% anniffiniedig
```

```
\DeclareRobustCommand{\elweight}{%% anniffiniedig ond rheolau
 \not@math@alphabet\elweight\relax
 \exfs@merge@weight{\eldefault}}% neu \exfs@try@series ?
\newcommand*{\uldefault}{ul}% anniffiniedig
\DeclareRobustCommand{\ulweight}{%% anniffiniedig ond rheolau
 \not@math@alphabet\ulweight\relax
 \exfs@merge@weight{\uldefault}}% neu \exfs@try@series ?
```

END added Original

`\dfshape` Something simpler.

```
\let\dfshape\normalshape
```

`\swashapedefault` cfr: be' i wneud am hwn?

```
\newcommand*{\swshapedefault}{\itdefault}
```

LaTeX ddim yn cynnwys `\swstyle` felly . . . ? | LaTeX doesn't include `\swstyle` so . . . ?

```
\ExplSyntaxOn
\hook_gput_code:nnn {begindocument}{.}
{% compatibility with original nfss \swshape
```

Note this doesn't affect `\swashstyle` or `\textswash`

`\swshape` Conditional definition. We overwrite the kernel's definition either way. The `compat` option determines only with what we overwrite it.

```
  \bool_if:NTF \g__exfs_compat_bool
  {
    \DeclareRobustCommand{\swshape}
    {
      \not@math@alphabet\swshape\relax
      \swstyle\fontshape\swshapedefault\selectfont
    }
    \PackageWarning{nfssext-cfr}{
      Overwriting ~ kernel ~ definition ~ of ~ \swshape \space (compat)
    }
  }{
    \DeclareRobustCommand \swshape
    {
      \not@math@alphabet\swshape\relax
      \exfs@swshape
    }
    \PackageWarning{nfssext-cfr}{
      Overwriting ~ kernel ~ definition ~ of ~ \swshape \space (new)
    }
  }
```

`\textin` Conditional definition.

```
  \@ifpackageloaded{hyperref}{
    \hook_gput_code:nnn { cmd/textin/before } { . }
    {
      \exfs@info{
        Note ~ that ~ '\protect\textin' ~ is ~ defined ~ by ~ hyperref.\MessageBreak
        Use ~ for ~ inferior ~ digits ~ will ~ yield ~ an\MessageBreak
        undefined ~ command ~ error ~ in ~ document ~ font ~ encodings.\MessageBreak
        Use ~ '\protect\textinf' ~ to ~ access ~ inferior ~ digits
      }
    }
  }{
    \DeclareTextFontCommand{\textin}{\infstyle}
  }

}
\ExplSyntaxOff
```

\textln \textos \textinf \textsu \textsi \textdf

The annotation 'anniffiniedig' indicates the macro is *not* defined by the kernel as of 2024.

```
\DeclareTextFontCommand{\textln}{\lnstyle}% anniffiniedig
\DeclareTextFontCommand{\textos}{\osstyle}% anniffiniedig
\DeclareTextFontCommand{\textinf}{\infstyle}% anniffiniedig
\DeclareTextFontCommand{\textsu}{\sustyle}% anniffiniedig
\DeclareTextFontCommand{\textsi}{\sishape}% anniffiniedig
\DeclareTextFontCommand{\textdf}{\dfshape}% anniffiniedig
```

`\textsw` is already defined on newer kernels with essentially the same meaning as nfssext originally gave it, so we remove the definition here. However, the redefinition of `\swshape` means `\textsw` is effectively redefined, so the kernel definition is only technically retained.

BEGIN added (cfr)

\textti \textlt \textof \textalt \textreg \emboss \textorn \textqt \textsh \texttm \texttv

Families

```
\DeclareTextFontCommand{\textti}{\tistyle}% anniffiniedig
\DeclareTextFontCommand{\textlt}{\ltstyle}% anniffiniedig
\DeclareTextFontCommand{\textof}{\ofstyle}% anniffiniedig     % open-face (or outline or blank) style
\DeclareTextFontCommand{\textalt}{\altstyle}% anniffiniedig     % alternative style
\DeclareTextFontCommand{\textreg}{\regstyle}% anniffiniedig     % 'regular' style
\DeclareTextFontCommand{\emboss}{\embossstyle}% anniffiniedig
\DeclareTextFontCommand{\textorn}{\ornamentalstyle}% anniffiniedig % intended primarily for decorative initials etc.
\DeclareTextFontCommand{\textqt}{\qtstyle}% anniffiniedig
\DeclareTextFontCommand{\textsh}{\shstyle}% anniffiniedig      % shadowed style
\DeclareTextFontCommand{\texttm}{\tmstyle}% anniffiniedig
\DeclareTextFontCommand{\texttv}{\tvstyle}% anniffiniedig
```

\textl \texto \textp \textt \textpl \textpo \texttl \textto

Families - figures

```
\DeclareTextFontCommand{\textl}{\lstyle}% anniffiniedig
\DeclareTextFontCommand{\texto}{\ostyle}% anniffiniedig
```

```
\DeclareTextFontCommand{\textp}{\pstyle}% anniffiniedig
\DeclareTextFontCommand{\textt}{\tstyle}% anniffiniedig
\DeclareTextFontCommand{\textpl}{\plstyle}% anniffiniedig
\DeclareTextFontCommand{\textpo}{\postyle}% anniffiniedig
\DeclareTextFontCommand{\texttl}{\tlstyle}% anniffiniedig
\DeclareTextFontCommand{\textto}{\tostyle}% anniffiniedig
```

\textol \textswash \textu \textscu \textui \textri

Shapes

```
\DeclareTextFontCommand{\textol}{\olshape}% anniffiniedig   % outline
\DeclareTextFontCommand{\textswash}{\swashstyle}% anniffiniedig  % an attempt to improve on \textsw
\DeclareTextFontCommand{\textu}{\ushape}% anniffiniedig       % be' yw hwn?! <_ underlined?
\DeclareTextFontCommand{\textscu}{\scushape}% anniffiniedig
\DeclareTextFontCommand{\textui}{\uishape}% anniffiniedig   % upright italic
\DeclareTextFontCommand{\textri}{\rishape}% anniffiniedig   % reverse italic
```

\textnw \textcd \textec \textuc \textet \textep \textex \textux \textrw

Widths

```
\DeclareTextFontCommand{\textnw}{\nwwidth}% anniffiniedig
\DeclareTextFontCommand{\textcd}{\cdwidth}% anniffiniedig
\DeclareTextFontCommand{\textec}{\ecwidth}% anniffiniedig
\DeclareTextFontCommand{\textuc}{\ucwidth}% anniffiniedig
\DeclareTextFontCommand{\textet}{\etwidth}% anniffiniedig
\DeclareTextFontCommand{\textep}{\epwidth}% anniffiniedig
\DeclareTextFontCommand{\textex}{\exwidth}% anniffiniedig
\DeclareTextFontCommand{\textux}{\uxwidth}% anniffiniedig
\DeclareTextFontCommand{\textrw}{\regwidth}% anniffiniedig
```

\textmb \textdb \textbd \textsb \texteb \textub \textlg \textel \textul

Weights

```
\DeclareTextFontCommand{\textmb}{\mbweight}% anniffiniedig
\DeclareTextFontCommand{\textdb}{\dbweight}% anniffiniedig
\DeclareTextFontCommand{\textbd}{\bdweight}% new?
\DeclareTextFontCommand{\textsb}{\sbweight}% anniffiniedig
\DeclareTextFontCommand{\texteb}{\ebweight}% anniffiniedig
\DeclareTextFontCommand{\textub}{\ubweight}% anniffiniedig
\DeclareTextFontCommand{\textlg}{\lgweight}% anniffiniedig
\DeclareTextFontCommand{\textel}{\elweight}% anniffiniedig
\DeclareTextFontCommand{\textul}{\ulweight}% anniffiniedig
```

END added

BEGIN patch font initialisation for Latin Modern

Stop redefinition of bold if using Latin Modern as `clm`. Kernel default only blocks redefinition for `lm`. Don't rely on cfr-lm internal macros as they may change without notice . . . . We don't need Dunhill, though, because it doesn't have bold of any kind. cfr-lm doesn't support using Quotation Sans as default or using e.g. serif as default sans, but there's nothing to stop somebody doing that so follow the kernel here even though it makes for a massive list[13].

[13] I know this will go off the page when typeset, but I have no idea whether I can safely insert line breaks into the patch and I shall scream if I break this again. (Pun fully intended.)

```
\patchcmd{\init@series@setup}{cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}{cmr,cmss,cmtt,lcmss,l
  \PackageWarning{nfssext-cfr}{%
    Patching font initialisation macro for serif.%
  }%
}{%
  \PackageWarning{nfssext-cfr}{%
    Failed to patch font initialisation macro for serif.%
  }%
}
\patchcmd{\init@series@setup}{{cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}}{{cmr,cmss,cmtt,lcms
  \PackageWarning{nfssext-cfr}{%
    Patching font initialisation macro for sans.%
  }%
}{%
  \PackageWarning{nfssext-cfr}{%
    Failed to patch font initialisation macro for sans.%
  }%
}
\patchcmd{\init@series@setup}{{cmr,cmss,cmtt,lcmss,lcmtt,lmr,lmss,lmtt}}{{cmr,cmss,cmtt,lcms
  \PackageWarning{nfssext-cfr}{%
    Patching font initialisation macro for typewriter.%
  }%
}{%
  \PackageWarning{nfssext-cfr}{%
    Failed to patch font initialisation macro for typewriter.%
  }%
}
```

END

6.3 NFSS

This code was written for the *old* New Font Selection Scheme (NFSS). It should not generally be loaded on current or recent kernels.

nfssext-cfr-nfss (*pkg.*)

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{svn-prov}
\ProvidesPackageSVN[\filebase-nfss.sty]{$Id: nfssext-cfr.dtx 10366 2024-09-18 14:25:21Z cfrees $}[v1.0 \revinfo{} specially mangled by cfr; based on 2003/03/14 v1.2 Experimental NFSS Extensions; for old NFSS]
\DefineFileInfoSVN
```

:cfr-added: use ifthen

```
\RequirePackage{ifthen}
```

:end-added

\exfs@tempa
\exfs@tempb
\exfs@tempf

```
\newcommand*{\exfs@tempa}{}
\newcommand*{\exfs@tempb}{}
```

:cfr-added: extra variable (`\exfs@tempf`)

```
\newcommand*{\exfs@tempf}{}
```

:end-added

`\exfs@try@family`

```
\newcommand*{\exfs@try@family}[2][]{%
  \let\exfs@tempa\relax
  \begingroup
    \fontfamily{#2}\try@load@fontshape
    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
      \edef\exfs@tempa{#1}%
      \ifx\exfs@tempa\@empty
        \PackageWarning{nfssext}{%
          Font family '\f@encoding/#2' not available\MessageBreak
          Ignoring font switch}%
      \else
        \PackageInfo{nfssext}{%
          Font family '\f@encoding/#2' not available\MessageBreak
          Font family '\f@encoding/#1' tried instead}%
        \exfs@try@family{#1}%
      \fi
    \else
      \gdef\exfs@tempa{\fontfamily{#2}\selectfont}%
    \fi
  \endgroup
  \exfs@tempa}
```

`\exfs@try@series` :cfr-added `\exfs@try@series`

```
\newcommand*{\exfs@try@series}[2][]{%
  \let\exfs@tempa\relax
  \begingroup
    \fontseries{#2}\try@load@fontshape
    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
      \edef\exfs@tempa{#1}%
      \ifx\exfs@tempa\@empty
        \PackageWarning{nfssext-cfr}{%
          Font series '\f@encoding/\f@family/#2' not available\MessageBreak
          Ignoring font switch}%
      \else
        \PackageInfo{nfssext-cfr}{%
          Font family '\f@encoding/\f@family/#2' not available\MessageBreak
          Font family '\f@encoding/\f@family/#1' tried instead}%
        \exfs@try@series{#1}%
      \fi
    \else
      \gdef\exfs@tempa{\fontseries{#2}\selectfont}%
    \fi
  \endgroup
  \exfs@tempa
}
```

:end-added

\exfs@get@base
\exfs@get@variants
\exfs@next
\exfs@shift
\exfs@first
\exfs@part
\exfs@second

```
\def\exfs@get@base#1#2#3#4\@nil{#1#2#3}
```

:cfr-added:more `\exfs@` commands (get@variants, next, shift, first, part, second)

```
\def\exfs@get@variants#1#2#3#4\@nil{#4}
\def\exfs@next#1#2\@nil{#1}
\def\exfs@shift#1#2\@nil{#2}
\def\exfs@first#1#2\@nil{#1}
\def\exfs@part#1#2\@nil{#2}
\def\exfs@second#1#2#3\@nil{#2}
```

:end-added

\lnstyle
\osstyle
\infstyle
\instyle
\sustyle
\swstyle

```
\DeclareRobustCommand{\lnstyle}{%
  \not@math@alphabet\lnstyle\relax
  \exfs@try@family[\expandafter\exfs@get@base\f@family\@nil]%
  {\expandafter\exfs@get@base\f@family\@nil x}%
}
\DeclareRobustCommand{\osstyle}{%
  \not@math@alphabet\osstyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil j}}
\DeclareRobustCommand{\instyle}{%
  \not@math@alphabet\instyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil 0}}
\DeclareRobustCommand{\sustyle}{%
  \not@math@alphabet\sustyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil 1}}
\DeclareRobustCommand{\swstyle}{%
  \not@math@alphabet\swstyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil w}}
```

\exfs@merge@families :cfr-added - merge families

```
\newcommand*\exfs@merge@families[1]{%
  \edef\exfs@tempf{#1}%
  \edef\tempa{\expandafter\exfs@get@variants\f@family\@nil}%
  \edef\tempo{2j}%
  \let\exfs@tempq\@empty
  \def\exfs@tempg{}%
  \newif\ifadded
    \addedfalse
```

check whether there are variants - if not just use the requested addition

```
  \ifx\tempa\@empty
    \edef\exfs@tempq{\exfs@tempf}%
    \addedtrue
  \else
    \gdef\set{0,1,2,a,d,e,f,h,j,l,p,q,s,t,v,w}% these are the variants to consider - the order here and in the font name is crucial
    \ifx\tempo\exfs@tempf
      \@for \xx:=\set \do {%
```

check whether there are variants left - if not set the ‘next variant’ to empty

```
      \ifx\tempa\@empty
        \let\exfs@tempn\@empty
      \else
```

o/w get the next variant

```
        \edef\exfs@tempn{\expandafter\exfs@next\tempa\@nil}%
      \fi
      \edef\tempt{2}%
      \edef\tempj{j}%
```

if the next variant is 2 or j, ignore it

```
      \ifx\exfs@tempn\tempt
        \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
      \fi
      \ifx\exfs@tempn\tempj % if the next variant is j, ignore it
        \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
      \fi
```

see if the current value is either 2 or j and add it if so and if needed

```
      \ifx\tempt\xx
        \edef\exfs@tempg{\exfs@tempg\xx}%
      \else
        \ifx\tempj\xx % if the current value is j, we’re done
          \edef\exfs@tempq{\exfs@tempg\xx\tempa}%
          \let\tempa\@empty
          \addedtrue
        \else
```

o/w see if the current value matches the next variant

```
          \ifx\xx\exfs@tempn
            \edef\exfs@tempg{\exfs@tempg\xx}%
            \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
          \fi
        \fi
      \fi
     }%
    \else
     \@for \xx:=\set \do {%
```

check whether there are variants left and, if not, add the addition if needed

```
      \ifx\tempa\@empty
        \ifadded
        \else
          \edef\exfs@tempq{\exfs@tempg\exfs@tempf}%
          \addedtrue
        \fi
      \else
```

o/w get the next variant

```
        \edef\exfs@tempn{\expandafter\exfs@next\tempa\@nil}%
```

if the new token equals the next variant, combine whatever is saved in `\exfs@tempg` with whatever remains in `\tempa`

```
          \ifx\exfs@tempn\exfs@tempf
            \edef\exfs@tempq{\exfs@tempg\tempa}%
            \addedtrue
            \let\tempa\@empty
          \else
```

o/w, if the current value matches the requested addition, add it in

```
            \ifx\exfs@tempf\xx
              \edef\exfs@tempq{\exfs@tempg\xx\tempa}%
              \addedtrue
              \let\tempa\@empty
            \else
```

o/w, if the current value matches the next variant, shift

```
              \ifx\exfs@tempn\xx
                \edef\exfs@tempg{\exfs@tempg\xx}%
                \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
              \fi
            \fi
          \fi
        \fi
      }%
    \fi
  \fi
  \ifx\exfs@tempq\@empty
    \PackageError{nfssext-cfr}{Something is wrong here. Ignoring font switching command.}{}%
  \else
    \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil \exfs@tempq}%
  \fi
}
```

`\pstyle`
`\ostyle`
`\postyle`
`\tistyle`

```
\DeclareRobustCommand{\pstyle}{%  proportional figures
  \not@math@alphabet\pstyle\relax
  \exfs@merge@families{2}}
\DeclareRobustCommand{\tistyle}{% titling/display
  \not@math@alphabet\tistyle\relax
  \exfs@merge@families{d}}
\DeclareRobustCommand{\ostyle}{%  oldstyle figures (cf. original osstyle above)
  \not@math@alphabet\ostyle\relax
  \exfs@merge@families{j}}
```

combined command for proportional oldstyle

```
\DeclareRobustCommand{\postyle}{%
  \not@math@alphabet\postyle\relax
  \exfs@merge@families{2j}}
```

\ltstyle \ofstyle \altstyle \regstyle \embossstyle \ornamentalstyle \swashstyle \shstyle \qtstyle

note that this command is for use when the light version is a separate family rather than a weight variant (e.g. when you've got light, light bold etc. as well as regular weights)

```
\DeclareRobustCommand{\ltstyle}{%
  \not@math@alphabet\ltstyle\relax
  \exfs@merge@families{l}}
```

let's hope there aren't any fonts with a light family *and* an outline/openface/blank version

```
\DeclareRobustCommand{\ofstyle}{%
  \not@math@alphabet\ofstyle\relax
  \exfs@merge@families{l}}
\DeclareRobustCommand{\altstyle}{% alternative style
  \not@math@alphabet\altstyle\relax
  \exfs@merge@families{a}}
\DeclareRobustCommand{\regstyle}{% ``regular'' style
  \not@math@alphabet\regstyle\relax
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil}}
\DeclareRobustCommand{\embossstyle}{%
  \not@math@alphabet\embossstyle\relax
  \exfs@merge@families{e}}
\DeclareRobustCommand{\ornamentalstyle}{% intended primarily for decorative initial fonts etc.
  \not@math@alphabet\ornamentalstyle\relax
  \exfs@merge@families{p}}
\DeclareRobustCommand{\qtstyle}{% quotation style (assumes sans)
  \not@math@alphabet\qtstyle\relax
  \sffamily
  \exfs@merge@families{q}}
\DeclareRobustCommand{\shstyle}{%
  \not@math@alphabet\shstyle\relax
  \exfs@merge@families{h}}
\DeclareRobustCommand{\swashstyle}{%   an attempt to improve on \swstyle
  \not@math@alphabet\swashstyle\relax
  \exfs@merge@families{w}}
```

\tmstyle \tvstyle

Macros to switch between monowidth and variable typewriter. These need to unmerge before merging. We need to unmerge sans as well as the other kind of typewriter.

```
\DeclareRobustCommand{\tmstyle}{%       monowidth typewriter
  \not@math@alphabet\tmstyle\relax
  \exfs@unmerge@families{s}%
  \exfs@unmerge@families{v}%
  \exfs@merge@families{t}}
\DeclareRobustCommand{\tvstyle}{%       variable width typewriter
  \not@math@alphabet\tvstyle\relax
  \exfs@unmerge@families{s}%
  \exfs@unmerge@families{t}%
  \exfs@merge@families{v}}
```

\exfs@unmerge@families :cfr-added - unmerge families

```
\newcounter{taken}%
\newcommand*\exfs@unmerge@families[1]{%
  \edef\exfs@tempf{#1}%
  \edef\tempa{\expandafter\exfs@get@variants\f@family\@nil}%
  \let\exfs@tempq\@empty
  \edef\exfs@tempg{}%
  \setcounter{taken}{0}%
```

check whether there are variants - if not do nothing

```
  \ifx\tempa\@empty
      \edef\exfs@tempq{}%
  \else
```

o/w go through the variants to find the one to delete

```
      \whiledo{\value{taken}<1}{%
```

get the next variant

```
          \edef\exfs@tempn{\expandafter\exfs@next\tempa\@nil}%
```

see if the next variant is the thing we seek and, if so, eliminate it

```
          \ifx\exfs@tempf\exfs@tempn
              \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
              \edef\exfs@tempq{\exfs@tempg\tempa}%
              \stepcounter{taken}%
```

o/w save the next variant and move on if any variants remain

```
          \else
              \edef\exfs@tempg{\exfs@tempg\exfs@tempn}%
              \edef\tempa{\expandafter\exfs@shift\tempa\@nil}%
              \ifx\tempa\@empty% if there are no variants left, we're done
                  \edef\exfs@tempq{\exfs@tempg}%
                  \stepcounter{taken}%
              \fi
          \fi
      }%
  \fi
  \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil \exfs@tempq}%
}
```

\tstyle
\lstyle

```
\DeclareRobustCommand{\tstyle}{%  tabular figures
  \not@math@alphabet\tstyle\relax
  \exfs@unmerge@families{2}}
\DeclareRobustCommand{\lstyle}{%  lining figures (cf. command above)
  \not@math@alphabet\lstyle\relax
  \exfs@unmerge@families{j}}
```

\tlstyle
\plstyle
\tostyle

make a combined command for tabular lining

```
\DeclareRobustCommand{\tlstyle}{%
  \lstyle\tstyle}
```

proportional lining

```
\DeclareRobustCommand{\plstyle}{%
  \lstyle\pstyle}
```

tabular oldstyle ?!

```
\DeclareRobustCommand{\tostyle}{%
  \ostyle\tstyle}
```

\sidefault
\sishape

:end-added si is italic sc

```
\newcommand*{\sidefault}{si}
\DeclareRobustCommand{\sishape}{%
  \not@math@alphabet\sishape\relax
  \fontshape\sidefault\selectfont}
```

\oldefault
\olshape
\scoldefault
\scolshape

:cfr-added - is this how outline shapes should be handled?

```
\newcommand*{\oldefault}{ol}
\DeclareRobustCommand{\olshape}{%
  \not@math@alphabet\olshape\relax
  \fontshape\oldefault\selectfont}
\newcommand*{\scoldefault}{scol}
\DeclareRobustCommand{\scolshape}{%
  \not@math@alphabet\scolshape\relax
  \fontshape\scoldefault\selectfont}
```

\udefault
\ushape
\scudefault
\scushape

:fudge

```
\newcommand*{\udefault}{u}
\DeclareRobustCommand{\ushape}{%
  \not@math@alphabet\ushape\relax
  \fontshape\udefault\selectfont}
\newcommand*{\scudefault}{su}
\DeclareRobustCommand{\scushape}{%
  \not@math@alphabet\scushape\relax
  \fontshape\scudefault\selectfont}
```

\uidefault
\uishape
\ridefault
\rishape

:upright italic

```
\newcommand*{\uidefault}{ui}
\DeclareRobustCommand{\uishape}{%
  \not@math@alphabet\uishape\relax
  \fontshape\uidefault\selectfont}
```

:can i do this for reverse italic?

```
\newcommand*{\ridefault}{ri}
\DeclareRobustCommand{\rishape}{%
  \not@math@alphabet\rishape\relax
  \fontshape\ridefault\selectfont}
```

:end-added

\exfs@merge@shape

```
\newcommand*{\exfs@merge@shape}[3]{%
  \edef\exfs@tempa{#1}%
  \edef\exfs@tempb{#2}%
  \ifx\f@shape\exfs@tempb
    \expandafter\ifx\csname\f@encoding/\f@family/\f@series/#3\endcsname\relax
    \else
      \edef\exfs@tempa{#3}%
    \fi
  \fi
  \fontshape{\exfs@tempa}\selectfont}
```

\exfs@font@width :cfr-added - merge width changes into series

```
\newcommand*{\exfs@font@width}{%
  \edef\exfs@tempf{\expandafter\exfs@first\f@series\@nil }%
  \edef\exfs@temppart{\expandafter\exfs@part\f@series\@nil }%
  \ifx\exfs@temppart\@empty
      \def\exfs@width{}
  \else
      \edef\exfs@temps{\expandafter\exfs@second\f@series\@nil }%
      \ifx\exfs@temps{b}
          \edef\exfs@width{\expandafter\exfs@part\exfs@temps\@nil }%
      \else
          \ifx\exfs@temps{l}
              \edef\exfs@width{\expandafter\exfs@part\exfs@temps\@nil }%
          \else
              \edef\exfs@width{\exfs@temppart}%
          \fi
      \fi
  \fi
  \exfs@width
}
```

\exfs@merge@width

```
\newcommand*{\exfs@merge@width}[1]{%
  \edef\exfs@temph{#1}%
  \edef\exfs@tempf{\expandafter\exfs@first\f@series\@nil }%
  \edef\exfs@temppart{\expandafter\exfs@part\f@series\@nil }%
  \def\tempb{b}%
  \def\templ{l}%
  \ifx\exfs@temppart\@empty
      \def\exfs@series{\expandafter\exfs@tempf\exfs@temph}%
  \else
      \edef\exfs@temps{\expandafter\exfs@second\f@series\@nil }%
      \ifx\exfs@temps\tempb
          \def\exfs@series{\expandafter\exfs@tempf\exfs@temps\exfs@temph}%
      \else
          \ifx\exfs@temps\templ
              \def\exfs@series{\expandafter\exfs@tempf\exfs@temps\exfs@temph}%
          \else
              \def\exfs@series{\expandafter\exfs@tempf\exfs@temph}%
          \fi
      \fi
  \fi
  \exfs@try@series{\exfs@series}%
}
%^^A \fontseries\exfs@series\selectfont}
```

\exfs@unmerge@width

```
\newcommand*{\exfs@unmerge@width}{%
  \edef\exfs@tempf{\expandafter\exfs@first\f@series\@nil }%
  \edef\exfs@temppart{\expandafter\exfs@part\f@series\@nil }%
  \def\tempb{b}%
  \def\templ{l}%
  \ifx\exfs@temppart\@empty
      \def\exfs@series{\expandafter\exfs@tempf}%
  \else
      \edef\exfs@temps{\expandafter\exfs@second\f@series\@nil }%
      \ifx\exfs@temps\tempb
          \def\exfs@series{\expandafter\exfs@tempf\exfs@temps}%
      \else
          \ifx\exfs@temps\templ
              \def\exfs@series{\expandafter\exfs@tempf\exfs@temps}%
          \else
              \def\exfs@series{\expandafter\exfs@tempf}%
          \fi
      \fi
  \fi
  \exfs@try@series{\exfs@series}%
}
```

\regwidth

```
\DeclareRobustCommand{\regwidth}{%
  \not@math@alphabet\regwidth\relax
  \exfs@unmerge@width}
```

\nwdefault
\nwwidth
\cddefault
\cdwidth
\ecdefault
\ecwidth
\ucdefault
\ucwidth

```
\newcommand*{\nwdefault}{c}
\DeclareRobustCommand{\nwwidth}{%
  \not@math@alphabet\nwwidth\relax
  \exfs@merge@width{\nwdefault}}
\newcommand*{\cddefault}{c}
\DeclareRobustCommand{\cdwidth}{%
  \not@math@alphabet\cdwidth\relax
  \exfs@merge@width{\cddefault}}
\newcommand*{\ecdefault}{ec}
\DeclareRobustCommand{\ecwidth}{%
  \not@math@alphabet\ecwidth\relax
  \exfs@merge@width{\ecdefault}}
\newcommand*{\ucdefault}{uc}
\DeclareRobustCommand{\ucwidth}{%
  \not@math@alphabet\ucwidth\relax
  \exfs@merge@width{\ucdefault}}
```

\etdefault
\etwidth
\epdefault
\epwidth
\exdefault
\exwidth
\uxdefault
\uxwidth

```
\newcommand*{\etdefault}{x}
\DeclareRobustCommand{\etwidth}{%
  \not@math@alphabet\etwidth\relax
  \exfs@merge@width{\etdefault}}
\newcommand*{\epdefault}{x}
\DeclareRobustCommand{\epwidth}{%
  \not@math@alphabet\epwidth\relax
  \exfs@merge@width{\epdefault}}
\newcommand*{\exdefault}{ex}
\DeclareRobustCommand{\exwidth}{%
  \not@math@alphabet\exwidth\relax
  \exfs@merge@width{\exdefault}}
\newcommand*{\uxdefault}{ux}
\DeclareRobustCommand{\uxwidth}{%
  \not@math@alphabet\uxwidth\relax
  \exfs@merge@width{\uxdefault}}
```

\exfs@merge@weight :cfr-added merge weight changes into series

```
\newcommand*{\exfs@merge@weight}[1]{%
  \edef\exfs@tempg{#1}%
  \edef\exfs@tempf{\expandafter\exfs@first\f@series\@nil }%
  \edef\exfs@temppart{\expandafter\exfs@part\f@series\@nil }%
  \def\templ{l}%
  \def\tempb{b}%
```

:case when there's no second part, so the single character must be the weight and should be replaced

```
  \ifx\exfs@temppart\@empty
      \def\exfs@series{\expandafter\exfs@tempg}%
```

:case when there's a second part

```
  \else
```

:get first character of second part

```
      \edef\exfs@temps{\expandafter\exfs@second\f@series\@nil }%
      \edef\exfs@tempw{\expandafter\exfs@part\exfs@temps\@nil }%
```

:is the first character b? if so, it is part of the weight and should be replaced

```
      \ifx\exfs@temps\tempb
          \def\exfs@series{\expandafter\exfs@tempg\exfs@tempw}%
      \else
```

:is the first character l? if so, it is part of the weight and should be replaced

```
          \ifx\exfs@temps\templ
              \def\exfs@series{\expandafter\exfs@tempg\exfs@tempw}%
          \else
```

:o/w the first character is part of the width and should be retained

```
       \def\exfs@series{\expandafter\exfs@tempg\exfs@temppart}%
     \fi
   \fi
 \fi

 \ifx\exfs@tempg\exfs@series
   \exfs@try@series{\exfs@series}%
 \else
   \exfs@try@series[\exfs@tempg]{\exfs@series}%  assume user wants to change weight even if this changes back to the default width
 \fi
}
%     \end{macrocode}
% \end{macro}
% \begin{macro}{\mbdefault,\mbweight,\bddefault,\bfweight,\bdweight}
%    \begin{macrocode}
\newcommand*{\mbdefault}{mb}
\DeclareRobustCommand{\mbweight}{%
 \not@math@alphabet\mbweight\relax
 \exfs@merge@weight{\mbdefault}}
```

\dbdefault \dbweight \sbdefault \sbweight \ebdefault \ebweight \ubdefault \ubweight

Heavy weights.

```
\newcommand*{\dbdefault}{db}
\DeclareRobustCommand{\dbweight}{%
 \not@math@alphabet\dbweight\relax
 \exfs@merge@weight{\dbdefault}}
\newcommand*{\sbdefault}{sb}
\DeclareRobustCommand{\sbweight}{%
 \not@math@alphabet\sbweight\relax
 \exfs@merge@weight{\sbdefault}}
\newcommand*{\ebdefault}{eb}
\DeclareRobustCommand{\ebweight}{%
 \not@math@alphabet\ebweight\relax
 \exfs@merge@weight{\ebdefault}}
\newcommand*{\ubdefault}{ub}
\DeclareRobustCommand{\ubweight}{%
 \not@math@alphabet\ubweight\relax
 \exfs@merge@weight{\ubdefault}}
\newcommand*{\lgdefault}{l}
```

\lgdefault \lgweight \eldefault \elweight \uldefault \ulweight

note - use this if light is a variant weight, rather than a separate family

```
\DeclareRobustCommand{\lgweight}{%
 \not@math@alphabet\lgweight\relax
 \exfs@merge@weight{\lgdefault}}
\newcommand*{\eldefault}{el}
\DeclareRobustCommand{\elweight}{%
 \not@math@alphabet\elweight\relax
 \exfs@merge@weight{\eldefault}}
\newcommand*{\uldefault}{ul}
\DeclareRobustCommand{\ulweight}{%
 \not@math@alphabet\ulweight\relax
 \exfs@merge@weight{\uldefault}}
```

:end-added

\itshape
\scshape
\upshape
\dfshape

redefinition

```
\DeclareRobustCommand{\itshape}{%
  \not@math@alphabet\itshape\mathit
  \exfs@merge@shape{\itdefault}{\scdefault}{\sidefault}}
```

original :cfr-altered: `\scshape`

```
\DeclareRobustCommand{\scshape}{%
  \not@math@alphabet\scshape\relax
  \def\tempu{u}%
  \def\tempo{ol}%
  \ifx\f@shape\tempu
   \exfs@merge@shape{\scdefault}{\udefault}{\scudefault}%
  \else
   \ifx\f@shape\tempo
  \exfs@merge@shape{\scdefault}{\oldefault}{\scoldefault}%
 \else
    \exfs@merge@shape{\scdefault}{\itdefault}{\sidefault}%
 \fi
  \fi
}
```

:end-altered

```
\DeclareRobustCommand{\upshape}{%
  \not@math@alphabet\upshape\relax
  \exfs@merge@shape{\updefault}{\sidefault}{\scdefault}}
\DeclareRobustCommand{\dfshape}{%
  \not@math@alphabet\dfshape\relax
  \fontshape\shapedefault\selectfont}
```

\swashapedefault
\swshape

```
\newcommand*{\swshapedefault}{\itdefault}
\DeclareRobustCommand{\swshape}{%
  \not@math@alphabet\swshape\relax
  \swstyle\fontshape\swshapedefault\selectfont}
```

\textln
\textos
\textin
\textsu
\textsi
\textdf
\textsw

```
\DeclareTextFontCommand{\textln}{\lnstyle}
\DeclareTextFontCommand{\textos}{\osstyle}
\DeclareTextFontCommand{\textin}{\instyle}
\DeclareTextFontCommand{\textsu}{\sustyle}
\DeclareTextFontCommand{\textsi}{\sishape}
\DeclareTextFontCommand{\textdf}{\dfshape}
\DeclareTextFontCommand{\textsw}{\swshape}
```

:cfr-added

\textti
\textlt
\textof
\textalt
\textreg
\emboss
\textorn
\textqt
\textsh
\texttm
\texttv

Families

```
\DeclareTextFontCommand{\textti}{\tistyle}
\DeclareTextFontCommand{\textlt}{\ltstyle}
\DeclareTextFontCommand{\textof}{\ofstyle}      % open-face (or outline or blank) style
\DeclareTextFontCommand{\textalt}{\altstyle}     % alternative style
\DeclareTextFontCommand{\textreg}{\regstyle}     % ‘‘regular’’ style
\DeclareTextFontCommand{\emboss}{\embossstyle}
\DeclareTextFontCommand{\textorn}{\ornamentalstyle} % intended primarily for decorative initials etc.
\DeclareTextFontCommand{\textqt}{\qtstyle}
\DeclareTextFontCommand{\textsh}{\shstyle}      % shadowed style
\DeclareTextFontCommand{\texttm}{\tmstyle}
\DeclareTextFontCommand{\texttv}{\tvstyle}
```

\textl
\texto
\textp
\textt
\textpl
\textpo
\texttl
\textto

Families - figures

```
\DeclareTextFontCommand{\textl}{\lstyle}
\DeclareTextFontCommand{\texto}{\ostyle}
\DeclareTextFontCommand{\textp}{\pstyle}
\DeclareTextFontCommand{\textt}{\tstyle}
\DeclareTextFontCommand{\textpl}{\plstyle}
\DeclareTextFontCommand{\textpo}{\postyle}
\DeclareTextFontCommand{\texttl}{\tlstyle}
\DeclareTextFontCommand{\textto}{\tostyle}
```

\textol
\textswash
\textu
\textscu
\textui
\textri

Shapes

```
\DeclareTextFontCommand{\textol}{\olshape}      % outline
\DeclareTextFontCommand{\textswash}{\swashstyle}  % an attempt to improve on \textsw
\DeclareTextFontCommand{\textu}{\ushape}
\DeclareTextFontCommand{\textscu}{\scushape}
\DeclareTextFontCommand{\textui}{\uishape}      % upright italic
\DeclareTextFontCommand{\textri}{\rishape}      % reverse italic
```

\textnw
\textcd
\textec
\textuc
\textet
\textep
\textex
\textux
\textrw

Widths

```
\DeclareTextFontCommand{\textnw}{\nwwidth}
\DeclareTextFontCommand{\textcd}{\cdwidth}
\DeclareTextFontCommand{\textec}{\ecwidth}
\DeclareTextFontCommand{\textuc}{\ucwidth}
\DeclareTextFontCommand{\textet}{\etwidth}
\DeclareTextFontCommand{\textep}{\epwidth}
\DeclareTextFontCommand{\textex}{\exwidth}
\DeclareTextFontCommand{\textux}{\uxwidth}
\DeclareTextFontCommand{\textrw}{\regwidth}
```

\textmb
\textdb
\textbd
\textsb
\texteb
\textub
\textlg
\textel
\textul

Weights

```
\DeclareTextFontCommand{\textmb}{\mbweight}
\DeclareTextFontCommand{\textdb}{\dbweight}
\DeclareTextFontCommand{\textsb}{\sbweight}
\DeclareTextFontCommand{\texteb}{\ebweight}
\DeclareTextFontCommand{\textub}{\ubweight}
```

```
\DeclareTextFontCommand{\textlg}{\lgweight}
\DeclareTextFontCommand{\textel}{\elweight}
\DeclareTextFontCommand{\textul}{\ulweight}
```

end-added

Change History

SVN6140

General: Fixes a bug which prevented `\tmstyle` and `\tvstyle` working correctly if the current font was not a serif family. (Especially problematic in Beamer where `\normalfont` cannot be used as a workaround, but annoying elsewhere.) 1

Provides something a bit closer to real documentation. 1

2008-10-26

General: First public release as part of cfr-lm. 1

2008-12-22

General: Updated version released standalone. 1

2010-07-17

General: There should be no changes for the end user except that in certain cases it is possible that line-breaks may be altered if microtype is in use due to the enhanced support included for variant font families. 12

`\Microtype@Hook`: Add microtype support for variants. 15

`\qtstyle`: Improve `\ofstyle`. . . . 25

v0.0

`nfssext-cfr`: Update for NNFSS. . . 12

v1.0

`compat`: Add option `compat`. More aggressive/backwards compatible with `compat`. 13

`debug`: Add option `debug`. 13

`\exfs@merge@width`: Do not depend on incorrect series names, which are no longer supported. 30

`\exfs@series@splitter`: Rewritten as kernel no longer supports erroneous `m`. 21

`force`: Add option `force`. Load old file/incompatible with `force`. . 13

`nfssext-cfr`: Behaviour depends on kernel version and options. On newer kernels, quite conservative/less compatible by default. Fully compatible on older kernels. 12

So nfssext-cfr merging is now limited to family, weight and width. 12

Split `nfssext-cfr.sty` into `nfssext-cfr{,-nfss,-nnfss}.sty`. 12

`nfssext-cfr-nnfss`: Conditionally override kernel rules affecting switches to upright/small-caps.italic etc. . 28

Unconditionally add a bunch of shape change rules for shapes unsupported by the kernel - I don't *think* these should be problematic: if the current or requested shape is unsupported by the kernel, surely it can't be problematic to support that shape? 28

`\swshape`: Conditionally overwrite `\swshape` to take account of default setting. This is not for any package I know of on CTAN, but the original code used `\textsw`. 33

`\ubweight`: Make `\mbdefault` sb (duplicating `\sbdefault`) as I can't come up with anything better. 32

v6140

General: Extend documentation somewhat. 12

`\tvstyle`: Modify `\tmstyle` and `\tvstyle` to unmerge sans and other typewriter before merging appropriate variant. 26

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.