NAME

autoinst – wrapper around the LCDF TypeTools, for installing and using OpenType fonts in LaTeX.

SYNOPSIS

autoinst [*options*] **font(s)**

DESCRIPTION

Eddie Kohler's *LCDF TypeTools* are superb tools for installing OpenType fonts in LaTeX, but they can be hard to use: they need many, often long, command lines and don't generate the *fd* and *sty* files LaTeX needs. **autoinst** simplifies the use of the *TypeTools* for font installation by generating and executing all commands for *otftotfm* and by creating and installing all necessary *fd* and *sty* files.

Given a family of font files (in *otf* or *ttf* format), **autoinst** will create several LaTeX font families:

- Four text families (with lining and oldstyle digits, each in both tabular and proportional variants), all with the following shapes:

n	Roman (i.e., upright) text
it, *sl*	Italic and slanted (sometimes called oblique) text
sc	Small caps
scit, *scsl*	Italic and slanted small caps
sw	Swash
nw	'Upright swash'

- For each T1–encoded text family: a family of TS1–encoded symbol fonts, in roman, italic and slanted shapes.
- Families with superiors, inferiors, numerators and denominators, in roman, italic and slanted shapes.
- Families with 'Titling' characters; these '... replace the default glyphs with corresponding forms designed specifically for titling. These may be all-capital and/or larger on the body, and adjusted for viewing at larger sizes' (according to the OpenType Specification).
- An ornament family, also in roman, italic and slanted shapes.

Of course, if your fonts don't contain italics, oldstyle digits, small caps etc., the corresponding shapes and families are not created. In addition, the creation of most families and shapes can be controlled by the user (see "COMMAND-LINE OPTIONS" below).

These families use the *FontPro* project's naming scheme: <*FontFamily*>–<*Suffix*>, where <*Suffix*> is:

LF	proportional (i.e., figures have varying widths) lining figures
TLF	tabular (i.e., all figures have the same width) lining figures
OsF	proportional oldstyle figures
TOsF	tabular oldstyle figures
Sup	superior characters (note that most fonts have only an incomplete set of superior characters: digits, some punctuation and the letters *abdeilmnorst*; normal forms are used for other characters)
Inf	inferior characters; usually only digits and some punctuation, normal forms for other characters
Titl	Titling characters; see above
Orn	ornaments
Numr	numerators
Dnom	denominators

The individual fonts are named <*FontName*>–<*suffix*>–<*shape*>–<*enc*>, where <*suffix*> is the same as above (but in lowercase), <*shape*> is either empty, 'sc' or 'swash', and <*enc*> is the encoding (also in lowercase). A typical name in this scheme would be 'FiraSans–Light–osf–sc–ly1'.

Using the fonts in your LaTeX documents

autoinst generates a style file for using the fonts in LaTeX documents, named *<FontFamily>.sty*. This style file also takes care of loading the *fontenc* and *textcomp* packages. To use the fonts, add the command `\usepackage{<FontFamily>}` to the preamble of your document.

This style file defines a number of options:

`mainfont`
: Redefine `\familydefault` to make this font the main font for the document. This is a no-op if the font is installed as a serif font; but if the font is installed as a sanserif or typewriter font, this option saves you from having to redefine `\familydefault` yourself.

`lining`, `oldstyle`, `tabular`, `proportional`
: Choose which figure style to use. The defaults are 'oldstyle' and 'proportional' (if available).

`scale=`*<number>*, `scale=MatchLowercase`
: Scale the font by a factor of *<number>*. E.g., to increase the size of the font by 5%, use `\usepackage[scale=1.05]{<FontFamily>}`. The special value `MatchLowercase` may be used to scale the font so that its x–height matches that of the previously active font (which is usually Computer Modern, unless you have loaded another font package before this one). The name `scaled` may be used as a synonym for `scale`.

`medium`, `book`, `text`, `regular`
: Select the weight that LaTeX will use as the 'regular' weight; the default is `regular`.

`heavy`, `black`, `extrabold`, `demibold`, `semibold`, `bold`
: Select the weight that LaTeX will use as the 'bold' weight; the default is `bold`.

The last two groups of options will only work if you have the *mweights* package installed.

The style file will also try to load the *fontaxes* package (on CTAN), which gives easy access to various font shapes and styles. Using the machinery set up by *fontaxes*, the generated style file defines a number of commands (which take the text to be typeset as argument) and declarations (which don't take arguments, but affect all text up to the end of the current group) to access titling, superior and inferior characters:

```
DECLARATION     COMMAND          SHORT FORM OF COMMAND

\tlshape        \texttitling     \texttl
\supfigures     \textsuperior    \textsup, \textsu
\inffigures     \textinferior    \textinf, \textin
```

In addition, the `\swshape` and `\textsw` commands are redefined to place swash on *fontaxes*' secondary shape axis (*fontaxes* places it on the primary shape axis) to make them behave properly when nested, so that `\swshape\upshape` will give upright swash.

There are no commands for accessing the numerator and denominator fonts; these can be selected using *fontaxes*' standard commands, e.g., `\fontfigurestyle{numerator}\selectfont`.

These commands are only generated for existing shapes and number styles; no commands are generated for shapes and styles that don't exist, or whose generation was turned off by the user. Also these commands are built on top of *fontaxes*, so if that package cannot be found, you're limited to using the lower-level commands from standard NFSS (`\fontfamily`, `\fontseries`, `\fontshape` etc.).

By default, **autoinst** generates text fonts with OT1, LY1 and T1 encodings, and the generated style files use T1 as the default text encoding. Other encodings can be chosen using the *−encoding* option (see "COMMAND-LINE OPTIONS" below).

Maths

This is an experimental feature; **USE AT YOUR OWN RISK!** Test the results thoroughly before using them in real documents, and be warned that future versions of **autoinst** may introduce incompatible changes.

The *−math* option tells **autoinst** to generate basic math fonts. When enabled, the generated style file defines a few extra options to access these math fonts:

`math`
: Use these fonts for the maths in your document.

`mathlining`, `matholdstyle`
: Choose which figure style to use in maths. The default is 'mathlining'.

`mathcal`
: Use the swash characters from your fonts as the `\mathcal` alphabet. (This option only exists if your fonts actually contain swash characters and a `swsh` feature to access them).

`math-style=<`*`style`*`>`
: Choose the 'math style' to use. With `math-style=ISO`, all latin and greek letters in math are italic; with `math-style=TeX` (the default), uppercase greek is upright; with `math-style=french`, all greek as well as uppercase latin is upright; and with `math-style=upright` all letters are upright.

Note that this 'math' option only changes digits, latin and greek letters, plus a few basic punctuation characters; all other mathematical symbols, operators, delimiters etc. are left as they were before. If you don't want to use TeX's default versions of those symbols, load another math package (such as *mathdesign* or *newtxmath*) before loading the **autoinst**–generated style file.

Finally, note that **autoinst** doesn't check if your fonts actually contains all of the required characters; it just assumes that they do and sets up the style file accordingly. Even if your fonts contain greek, characters such as `\varepsilon` may be missing. You may also find that some glyphs *are* present in your fonts, but don't work well in equations or don't match with other symbols; edit the generated style file to remove the declarations of these offending characters. Once again: test the results before using them! If the characters themselves are fine but spaced too tightly, you may try increasing the side bearings in math fonts with the *–mathspacing* option (see below), e.g. `–mathspacing=100`.

NFSS codes

LaTeX's New Font Selection System (NFSS) identifies fonts by a combination of family, series (the concatenation of weight and width), shape and size. **autoinst** parses the font's metadata to determine these parameters. When this fails (usually because the font family contains uncommon weights, widths or shapes), **autoinst** ends up with different fonts having the *same* values for these font parameters; such fonts cannot be used in NFSS, since there's no way distinguish them. When **autoinst** detects such a situation, it will print an error message and abort. If that happens, either rerun **autoinst** on a smaller set of fonts, or add the missing widths, weights and shapes to the tables `WIDTH`, `WEIGHT` and `SHAPE` in the source code. Please also send a bug report (see AUTHOR below).

The mapping of shapes to NFSS codes is done using the following table:

```
SHAPE                              CODE
-------------------------------    ----
Roman, Upright                     n
Italic                             it
Oblique, Slant(ed), Incline(d)     sl
```

(*Exception:* Adobe Silentium Pro contains two Roman shapes; we map the first of these to 'n', for the second one we (ab)use the 'it' code as this family doesn't contain an Italic shape.)

The mapping of weights and widths to NFSS codes is a more complex, two-step proces. In the first step, all fonts are assigned a 'series' name that is simply the concatenation of its weight and width (after expanding any abbreviations and converting to lowercase). A font with 'Cond' width and 'Ultra' weight will then be known as 'ultrablackcondensed'.

In the second step, **autoinst** tries to map all combinations of NFSS codes (ul, el, l, sl, m, sb, b, eb and ub for weights; uc, ec, c, sc, m, sx, x, ex and ux for widths) to actual fonts. Of course, not all 81 combinations of these NFSS weights and widths will map to existing fonts; and conversely it may not be possible to assign every existing font a unique code in a sane way (especially for the weights, some font families offer more choices or finer granularity than NFSS's codes can handle; e.g., Fira Sans contains fifteen(!) different weights, including an additional 'Medium' weight between Regular and Semibold).

autoinst tries hard to ensure that the most common NFSS codes (and high-level commands such as

`\bfseries`, which are built on top of those codes) will 'just work'.

To see exactly which NFSS codes map to which fonts, see the log file (pro tip: run **autoinst** with the *−dryrun* option to check the chosen mapping beforehand). The *−nfssweight* and *−nfsswidth* command-line options can be used to finetune the mapping between NFSS codes and fonts.

To access specific weights or widths, one can always use the `\fontseries` command with the full series name (i.e., `\fontseries{demibold}\selectfont`).

Ornaments

Ornament fonts are regular LY1−encoded fonts, with a number of 'regular' characters replaced by ornament glyphs. The OpenType specification says that fonts should only put their ornaments in place of the lowercase ASCII letters, but some fonts put them in other positions (such as those of the digits) as well.

Ornaments can be accessed like `{\ornaments a}` and `{\ornaments\char"61}`, or equivalently `\textornaments{a}` and `\textornaments{\char"61}`. To see which ornaments a font contains (and at which positions), run LaTeX on the file *nfssfont.tex* (which is included in any standard LaTeX installation), supply the name of the ornament font (i.e., `GaramondLibre-Regular-orn-u`) and give the command `\table\bye`; this will create a table of all glyphs in that font.

Note that versions of **autoinst** up to 20200428 handled ornaments differently, and fonts and style files generated by those versions are not compatible with files generated by newer versions.

COMMAND-LINE OPTIONS

autoinst tries hard to do The Right Thing (TM) by default, so you usually won't need these options; but most aspects of its operation can be fine-tuned if you want to.

You may use either one or two dashes before options, and option names may be shortened to a unique prefix (e.g., **−encoding** may be abbreviated to **−enc** or even **−en**, but **−e** is ambiguous (it may mean either **−encoding** or **−extra**)).

−version
: Print **autoinst**'s version number and exit.

−help
: Print a (relatively) short help text and exit.

−dryrun
: Don't generate output; just parse input fonts and write a log file saying what **autoinst** would have done.

−logfile=*filename*
: Write log data to *filename* instead of the default *<fontfamily>.log*. If the file already exists, **autoinst** appends to it; it doesn't overwrite an existing file.

−verbose
: Add more details to the log file.

−encoding=*encoding[,encoding]*
: Generate the specified encoding(s) for the text fonts. Multiple encodings may be specified as a comma-separated list (without spaces!); the default choice of encodings is 'OT1,LY1,T1'.

  For each specified encoding XYZ, **autoinst** will first see if there is an encoding file *XYZ.enc* in the current directory, and if found it will use that; otherwise it will use one of its built-in encoding files. Currently **autoinst** comes with support for the OT1, T1/TS1, LY1, LGR, T2A/B/C and T3/TS3 encodings. (These files are called *fontools_ot1.enc* etc. to avoid name clashes with other packages; the 'fontools_' prefix may be omitted.)

−ts1/−nots1
: Control the creation of TS1−encoded fonts. The default is **−ts1** if the text encodings (see *−encoding* above) include T1, **−nots1** otherwise.

–serif/–sanserif/–typewriter
Install the font as a serif, sanserif or typewriter font, respectively. This changes how you access the font in LaTeX: with `\rmfamily`/`\textrm`, `\sffamily`/`\textsf` or `\ttfamily`/`\texttt`.

Installing the font as a typewriter font will cause two further changes: it will – by default – turn off the use of f–ligatures (though this can be overridden with the *–ligatures* option), and it will disable hyphenation for this font. This latter effect cannot be re-enabled in **autoinst**; if you want typewriter text to be hyphenated, use the *hyphenat* package.

If none of these options is specified, **autoinst** tries to guess: if the font's filename contains the string 'mono' or if the field `isFixedPitch` in the font's *post* table is True, it will select **–typewriter**; else if the filename contains 'sans' it will select **–sanserif**; otherwise it will opt for **–serif**.

–lining/–nolining
Control the creation of fonts with lining figures. The default is **–lining**.

–oldstyle/–nooldstyle
Control the creation of fonts with oldstyle figures. The default is **–oldstyle**.

–proportional/–noproportional
Control the creation of fonts with proportional figures. The default is **–proportional**.

–tabular/–notabular
Control the creation of fonts with tabular figures. The default is **–tabular**.

–smallcaps/–nosmallcaps
Control the creation of small caps fonts. The default is **–smallcaps**.

–swash/–noswash
Control the creation of swash fonts. The default is **–swash**.

–titling/–notitling
Control the creation of titling families. The default is **–titling**.

–superiors/–nosuperiors
Control the creation of fonts with superior characters. The default is **–superiors**.

–noinferiors
–inferiors [= **none** | **auto** | **subs** | **sinf** | **dnom**]
The OpenType standard defines several kinds of digits that might be used as inferiors or subscripts: 'Subscripts' (OpenType feature 'subs'), 'Scientific Inferiors' ('sinf'), and 'Denominators' ('dnom'). This option allows the user to determine which of these styles **autoinst** should use for the inferior characters. Alternatively, the value 'auto' tells **autoinst** to use the first value in 'sinf', 'subs' or 'dnom' that is supported by the font. Saying just **–inferiors** is equivalent to **–inferiors=auto**; otherwise the default is **–noinferiors**.

If you specify a style of inferiors that isn't present in the font, ***autoinst*** *will fall back to its default behaviour of not creating fonts with inferiors at all; it won't try to substitute one of the other styles.*

–fractions/–nofractions
Control the creation of fonts with numerators and denominators. The default is **–nofractions**.

–ornaments/–noornaments
Control the creation of ornament fonts. The default is **–ornaments**.

–ligatures/–noligatures
Some fonts create glyphs for the standard f–ligatures (ff, fi, fl, ffi, ffl), but don't provide a 'liga' feature to access these. This option tells **autoinst** to add extra `LIGKERN` rules to the generated fonts to enable the use of these ligatures. The default is **–ligatures**, unless the user specified the *–typewriter* option.

Specify **–noligatures** to disable the generation of ligatures even for fonts that do contain a 'liga' feature.

−math
Tells **autoinst** to create basic math fonts (see above).

−mathspacing=*amount*
Letterspace each character in the math fonts by *amount* units, where 1000 units equal one em. In my opinion, many text fonts benefit from letterspacing by 50 to 100 units when used in maths; some fonts need even more. Use your own judgement!

−defaultlining/−defaultoldstyle
−defaulttabular/−defaultproportional
Tell **autoinst** which figure style is the current font family's default (i.e., which figures you get when you don't specify any OpenType features).

Don't use these options unless you are certain you need them! They are only needed for fonts that don't provide OpenType features for their default figure style; and even in that case, **autoinst**'s default values (**−defaultlining** and **−defaulttabular**) are usually correct.

−nofigurekern
Some fonts provide kerning pairs for tabular figures. This is very probably not what you want (e.g., numbers in tables won't line up exactly). This option adds extra *−−ligkern* options to the commands for *otftotfm* to suppress such kerns. Note that this option leads to very long commands (it adds one hundred *−−ligkern* options), which may cause problems on some systems.

−nfssweight=*code*=*weight*
−nfsswidth=*code*=*width*
Map the NFSS code *code* to the given weight or width, overriding the built-in tables. Each of these options may be given multiple times, to override more than one NFSS code. Example: to map the 'ul' code to the 'Thin' weight, use `-nfssweight=ul=thin`. To inhibit the use of the 'ul' code completely, use `-nfssweight=ul=`.

−extra=*extra*
Append *extra* as extra options to the command lines for *otftotfm*. To prevent *extra* from accidentily being interpreted as options to **autoinst**, it should be properly quoted.

−manual
Manual mode; for users who want to post-process the generated files and commands. By default, **autoinst** immediately executes all *otftotfm* commands it generates; in manual mode, these are instead written to a file *autoinst.bat*. Furthermore it tells *otftotfm* to generate human readable (and editable) *pl/vpl* files instead of the default *tfm/vf* ones, and to place all generated files in a subdirectory `./autoinst_output/` of the current directory, rather than install them into your TeX installation.

When using this option, you need to execute the following manual steps after **autoinst** has finished:

− run *pltotf* and *vptovf* on the generated *pl* and *vf* files, to convert them to *tfm/vf* format;
− move all generated files to a proper TEXMF tree, and, if necessary, update the filename database;
− tell TeX about the new *map* file (usually by running `updmap` or similar).

Note that some options (*−target*, *−vendor* and *−typeface*, *−[no]updmap*) are meaningless, and hence ignored, in manual mode.

−target=*DIRECTORY*
Install all generated files into the TEXMF tree at *DIRECTORY*.

By default, **autoinst** searches the `$TEXMFLOCAL` and `$TEXMFHOME` trees and installs all files into the first user-writable TEXMF tree it finds. If **autoinst** cannot find such a user-writable directory (which shouldn't happen, since `$TEXMFHOME` is supposed to be user-writable) it will print a warning message and put all files into the subdirectory `./autoinst_output/` of the current directory. It's then up to the user to move the generated files to a better location and update all relevant databases (usually by calling *texhash* and *updmap*).

WARNING: using this option may interfere with *kpathsea* and *updmap* (especially when the chosen directory is outside the standard TEXMF trees), so using *−target* will disable the automatic call to

updmap (as if *−noupdmap* had been given). It is up to the user to manually update all databases (i.e., by calling *texhash* and *updmap* or similar).

−vendor=*VENDOR*

−typeface=*TYPEFACE*

These options are equivalent to *otftotfm*'s *−−vendor* and *−−typeface* options: they change the 'vendor' and 'typeface' parts of the names of the subdirectories in the TEXMF tree where generated files will be stored. The default values are 'lcdftools' and the font's FontFamily name.

Note that these options change *only* directory names, not the names of any generated files.

−updmap/−noupdmap

Control whether or not *updmap* is called after the last call to *otftotfm*. The default is **−updmap**.

A note for MiKTeX users

Automatically installing the fonts into a suitable TEXMF tree (as **autoinst** tries to do by default) only works for TeX-installations that use the *kpathsea* library; with TeX distributions that implement their own directory searching (such as MiKTeX), **autoinst** will complain that it cannot find the *kpsewhich* program and move all generated files into a subdirectory `./autoinst_output/` of the current directory. If you use such a TeX distribution, you should either move these files to their correct destinations by hand, or use the *−target* option (see "COMMAND-LINE OPTIONS" below) to manually specify a TEXMF tree.

Also, some OpenType fonts contain so many kerning pairs that the resulting *pl* and *vpl* files are too big for MiKTeX's *pltotf* and *vptovf*; the versions that come with W32TeX (*http://www.w32tex.org*) and TeXLive (*http://tug.org/texlive*) don't seem to have this problem.

A note for MacTeX users

By default, **autoinst** will try to install all generated files into the `$TEXMFLOCAL` tree; when this directory isn't user-writable, it will use the `$TEXMFHOME` tree instead. Unfortunately, MacTeX's version of `updmap-sys` (which is called behind the scenes) doesn't search in `$TEXMFHOME`, and hence MacTeX will not find the new fonts.

To remedy this, either run **autoinst** as root (so that it can install everything into `$TEXMFLOCAL`) or manually run `updmap -user` to tell TeX about the files in `$TEXMFHOME`. The latter option does, however, have some caveats; see *https://tug.org/texlive/scripts−sys−user.html*.

SEE ALSO

Eddie Kohler's **TypeTools** (*http://www.lcdf.org/type*).

Perl can be obtained from *http://www.perl.org*; it is included in most Linux distributions. For Windows, try ActivePerl (*http://www.activestate.com*) or Strawberry Perl (*http://strawberryperl.com*).

XeTeX (*http://www.tug.org/xetex*) and **LuaTeX** (*http://www.luatex.org*) are Unicode-aware TeX engines that can use OpenType fonts directly, without any (La)TeX−specific support files.

The **FontPro** project (*https://github.com/sebschub/FontPro*) offers very complete LaTeX support (even for typesetting maths) for Adobe's Minion Pro, Myriad Pro and Cronos Pro font families.

AUTHOR

Marc Penninga (*marcpenninga@gmail.com*)

When sending a bug report, please give as much relevant information as possible; this usually includes (but may not be limited to) the log file (please add the *−verbose* command-line option, for extra info). If you see any error messages, please include these *verbatim*; don't paraphase.

COPYRIGHT

LICENSE

DISCLAIMER

VERSION

This document describes **autoinst** version 20200527.

RECENT CHANGES

(See the source for the full story, all the way back to 2005.)

2020−05−27 Added basic (and still somewhat experimental) math support. Implemented the `scale=MatchLowercase` option value in the generated style files. 'Wide' fonts are mapped to the 'sx' NFSS code instead of 'x', to cater for League Mono Variable's Wide and Extended widths. The generated style files now use `\textsup` and `\textinf` instead of the more cryptic `\textsu` and `\textin` to access superior and inferior characters (though the old forms are retained for backwards compatibility).

2020−05−11 When present, use encoding files in the current working directory in preference of the ones that come with **autoinst**. Changed the way ornament fonts are created; ornament glyphs are now always included in the position chosen by the font's designer.

2020−04−28 Fix a bug where the first font argument would be mistaken for an argument to *−inferiors*.

2020−01−29 Don't create empty subdirectories in the target TEXMF tree.

2019−11−18 Fine-tuned calling of *kpsewhich* on Windows (patch by Akira Kakuto). The font info parsing now also recognises numerical weights, e.g. in Museo.

2019−10−29 The generated style files now use T1 as the default text encoding.

2019−10−27 The mapping in *fd* files between font series and standard NFSS attributes now uses the new *alias* function instead of *ssub* (based on code by Frank Mittelbach). The way *otftotfm* is called was changed to work around a Perl/Windows bug; the old way might cause the process to hang. Using the *−target* option now implies *−noupdmap*, since choosing a non-standard target directory interferes with kpathsea/texhash and updmap.

2019−10−01 Handle *−target* directories with spaces in their path names. Tweaked messages and logs to make them more useful to the user.

2019−07−12 Replaced single quotes in calls to *otfinfo* with double quotes, as they caused problems on Windows 10.

2019−06−25

- Added the *−mergeweights* and *−mergeshapes* options, and improved *−mergewidths*.
- Improved the parsing of fonts' widths and weights.
- Improved the mapping of widths and weights to NFSS codes.
- Changed logging code so that that results of font info parsing are always logged, even (especially!) when parsing fails.
- Added a warning when installing fonts from multiple families.
- Added simple recognition for sanserif and typewriter fonts.
- Fixed error checking after calls to *otfinfo* (**autoinst** previously only checked whether `fork()` was successful, not whether the actual call to *otfinfo* worked).
- Fixed a bug in the *−inferiors* option; when used without a (supposedly optional) value, it would silently gobble the next option instead.

2019−05−22 Added the *mainfont* option to the generated *sty* files. Prevented hyphenation for typewriter fonts (added `\hyphenchar\font=-1` to the `\DeclareFontFamily` declarations). Added the *−version* option.

2019−05−17 Changed the way the *−ligatures* option works: *−ligatures* enables f−ligatures (even without a ‘liga’ feature), *−noligatures* now disables f−ligatures (overriding a ‘liga’ feature).

2019−05−11 Separate small caps families are now also recognised when the family name ends with ‘SC’ (previously **autoinst** only looked for ‘SmallCaps’).

2019−04−22 Fixed a bug in the generation of swash shapes.

2019−04−19 Fixed a bug that affected −mergesmallcaps with multiple encodings.

2019−04−16 Added the <−mergesmallcaps> option, to handle cases where the small caps fonts are in separate font families. Titling shape is now treated as a separate family instead of a distinct shape; it is generated only for fonts with the ‘titl’ feature. Only add f−ligatures to fonts when explicitly asked to (*−ligatures*).

2019−04−11 Tried to make the log file more relevant. Added the *−nfssweight* and *−nfsswidth* options, and finetuned the automatic mapping between fonts and NFSS codes. Changed the name of the generated log file to *<fontfamily>.log*, and revived the *−logfile* option to allow overriding this choice. Made *−mergewidths* the default (instead of *−nomergewidths*).

2019−04−01 Fine-tuned the decision where to put generated files; in particular, create `$TEXMFHOME` if it doesn’t already exist and `$TEXMFLOCAL` isn’t user-writable.

In manual mode, or when we can’t find a user-writable TEXMF tree, put all generated files into a subdirectory `./autoinst_output/` instead of all over the current working directory.

Added ‘auto’ value to the *inferiors* option, to tell **autoinst** to use whatever inferior characters are available.

2019−03−14 Overhauled the mapping of fonts (more specifically of weights and widths; the mapping of shapes didn’t change) to NFSS codes. Instead of inventing our own codes to deal with every possible weight and width out there, we now create ‘long’ codes based on the names in the font metadata. Then we add ‘ssub’ rules to the *fd* files to map the standard NFSS codes to our fancy names (see the section **NFSS codes**; based on discussions with Frank Mittelbach and Bob Tennent).