

# La elección tipográfica en el sistema PYST-PUB

## El recorrido completo de font=<perfil> a \setmathfont: los perfiles ordinarios y el caso font=minion

Calude Code y F. Javier Payán Somet

19 de septiembre de 2026

Versión 2.3 de la nota

### Índice

|          |                                                                    |          |
|----------|--------------------------------------------------------------------|----------|
| <b>I</b> | <b>La elección tipográfica ordinaria en PYST-PUB</b>               | <b>4</b> |
| <b>1</b> | <b>Presentación</b>                                                | <b>4</b> |
| 1.1      | Los ficheros implicados . . . . .                                  | 4        |
| <b>2</b> | <b>De font= a \g_pyst_font_tl</b>                                  | <b>5</b> |
| <b>3</b> | <b>pyst-fonts.ltx: el orquestador de la carga</b>                  | <b>5</b> |
| 3.1      | El registro de nombres, por si no se carga ningún perfil . . . . . | 5        |
| 3.2      | El declarador se carga antes que cualquier perfil . . . . .        | 6        |
| 3.3      | El catálogo de fuentes válidas, construido por barrido . . . . .   | 6        |
| 3.4      | Decidir primero, cargar después . . . . .                          | 7        |
| <b>4</b> | <b>El declarador: \PystDeclareFontProfile</b>                      | <b>8</b> |
| 4.1      | Anatomía de la llamada: las claves . . . . .                       | 9        |
| 4.2      | La validación previa: \pyst_fp_check: . . . . .                    | 10       |
| 4.3      | fontspec siempre primero . . . . .                                 | 11       |

---

|           |                                                                                        |           |
|-----------|----------------------------------------------------------------------------------------|-----------|
| 4.4       | Paso 4, decidir la ruta matemática . . . . .                                           | 11        |
| 4.5       | Dos condicionales anidados . . . . .                                                   | 14        |
| 4.6       | <code>\setup_text::</code> las tres familias de texto . . . . .                        | 15        |
| 4.7       | Paso 5b, el <code>\setmathfont</code> , después de <code>\setmainfont</code> . . . . . | 15        |
| 4.8       | Las familias de visualización . . . . .                                                | 17        |
| 4.9       | El registro de nombres legibles . . . . .                                              | 17        |
| <b>5</b>  | <b>El anclaje por nombre de fichero</b>                                                | <b>18</b> |
| <b>6</b>  | <b><code>pyst-fonts.ltx</code> §5: el repliegue fijo de símbolos</b>                   | <b>18</b> |
| <b>7</b>  | <b>Qué se ve desde fuera: el informe de depuración</b>                                 | <b>19</b> |
| <b>8</b>  | <b>Las 24 correspondencias perfil → fuente matemática</b>                              | <b>20</b> |
| <br>      |                                                                                        |           |
| <b>II</b> | <b>El caso <code>font=minion</code></b>                                                | <b>22</b> |
| <b>9</b>  | <b>Por qué Minion no es declarativo del todo</b>                                       | <b>22</b> |
| <b>10</b> | <b>Matemáticas antes de <code>fontspec</code></b>                                      | <b>22</b> |
| 10.1      | El repliegue, y un efecto secundario que conviene conocer . . . . .                    | 24        |
| <b>11</b> | <b>Familias de texto manuales, con tamaños ópticos</b>                                 | <b>24</b> |
| <b>12</b> | <b>Lo que Minion sí comparte con los demás perfiles</b>                                | <b>25</b> |
| <b>13</b> | <b>Numeración de ecuaciones con cifras de altura uniforme</b>                          | <b>26</b> |
| <b>14</b> | <b>Los otros dos sitios que consultan <code>\ifPystMinion</code></b>                   | <b>26</b> |
| 14.1      | En qué se parece y en qué difiere del camino ordinario . . . . .                       | 27        |

ESTA NOTA describe, paso a paso, todo lo que ocurre entre que un documento declara `font=<perfil>` y el momento en el que L<sup>A</sup>T<sub>E</sub>X tiene ya establecidas las tres familias de texto, las diez familias de visualización y la fuente matemática que va a usar en dicho documento.

Conviene señalar, como puede observarse en la autoría publicada, que esta documentación ha sido elaborada fundamentalmente por Claude<sup>1</sup>, que ha seguido el estilo desarrollado por el otro autor en una nota técnica sobre la fase PRE. La propia codificación de la elección tipográfica, a partir de la versión 2.0 del sistema PYST-PUB, ha sido desarrollada en sus elementos técnicos más avanzados por Claude, convirtiendo la codificación previa escrita por F. Javier Payán Somet en régimen L<sup>A</sup>T<sub>E</sub>X<sub>2</sub>ε en una versión en régimen `expl3` más compacta y, tal vez, más obscura, de ahí la necesidad de este documento para comprenderla. En el fichero `pyst-fontprofile.ltx` se comentan las razones de esta conversión.

Por último, en la actualidad el catálogo de tipografías suministrado por PYST-PUB tiene 25 perfiles, pero existen dos mundos separados desde el primer condicional que se examina ([Apartado 2](#)), y esta nota respeta esa separación en su propia estructura: la [Parte I](#) recorre el camino común a los *24 perfiles ordinarios* y la [Parte II](#) recorre, con el mismo detalle, el camino propio de `font=minion`, la única tipografía comercial del catálogo<sup>2</sup> y la única que utiliza una codificación diferente.

---

<sup>1</sup>Existen claras incorrecciones gramaticales que han podido pasar desapercibidas.

<sup>2</sup>Este segundo esquema, que mezcla tipografías .otf y Type 1, es autoría de F. Javier Payán Somet, que ha utilizado un conjunto notable de herramientas de software libre para realizar esta codificación. Lo mismo puede hacerse para un grupo muy numerosos de fuentes comerciales.

## Parte I

# La elección tipográfica ordinaria en PYST-PUB

### 1 Presentación

La elección tipográfica pertenece a la fase POST del sistema PYST-PUB, la que se ejecuta *después* de que la fase PRE, descrita en la nota técnica que lleva ese nombre, haya terminado su trabajo y haya dejado resuelto, entre otras cosas, el valor de la clave `font`. Aquí no se discute cómo se llega a ese valor; se toma como dato de partida y se sigue su recorrido hacia delante.

Como en el resto del sistema PYST-PUB, aquí también rige la misma regla de diseño que gobierna toda la fase PRE:

**No reinventar lo que `fontspec` y `unicode-math` ya hacen.**

El sistema PYST-PUB no sabe nada de OpenType, no calcula `Scale`, no decide qué es una familia matemática válida. Lo único que hace es *decidir*, a partir de lo que declara cada perfil, con qué argumentos exactos llamar a `\setmainfont`, `\setsansfont`, `\setmonofont`, `\newfontface` y `\setmathfont`, y en qué orden. Todo el trabajo real lo hacen esos cinco comandos; PYST-PUB sólo reúne los argumentos correctos y se los entrega.

Esa entrega tiene, sin embargo, una dificultad concreta desde la versión 2.3: *la fuente matemática* no siempre puede tomarse tal cual la declara el perfil. Puede que la fuente OpenType que el perfil prefiere no esté instalada en la máquina donde se compila, y el sistema necesita un plan de repliegue, de hecho, dos, que no rompa la compilación. La mayor parte de esta nota gira alrededor de ese enrutado y de sus cinco salidas posibles.

#### 1.1 Los ficheros implicados

La elección tipográfica está repartida en los ficheros de la [Tabla 1](#).

Tabla 1: Ficheros que intervienen en la elección de una fuente.

| Fichero                                              | Papel                                                                                                                                                                                                                                                        |
|------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>core/pre/pyst-build.ltx</code>                 | Fase PRE: exporta <code>font=</code> a la variable global <code>\g_pyst_font_tl</code> y activa <code>\ifPystMinion</code> si el valor es exactamente <code>minion</code> .                                                                                  |
| <code>core/post/pyst-fonts.ltx</code>                | Orquestador: lee <code>\g_pyst_font_tl</code> , construye el catálogo de fuentes válidas, decide si el perfil pedido existe y, si no, aplica el repliegue a <code>stixtwo</code> . Su §5 añade, al final, un repliegue <i>fijo</i> de símbolos (Apartado 6). |
| <code>defs/pyst-fontprofile.def</code>               | Declarador: define <code>\PystDeclareFontProfile</code> , la forma en la que se escribe cada perfil, y la ruta en la que se decide cómo se resuelven las matemáticas ( <code>\pyst_fp_math_decide:</code> ).                                                 |
| <code>defs/fonts/pyst-font-&lt;nombre&gt;.def</code> | Datos: 24 ficheros, uno por perfil, cada uno una única llamada a <code>\PystDeclareFontProfile</code> con sus seis u ocho decisiones reales.                                                                                                                 |

## 2 De font= a \g\_pyst\_font\_tl

Lo único que necesitamos saber de la fase PRE, a efectos de esta nota, es que deja el valor de la clave `font`, ya resuelto entre valores por defecto, tipo de documento y usuario, en la variable global `\g_pyst_font_tl`. Y que, de paso, activa un interruptor si ese valor es exactamente la cadena `minion`:

```
\str_if_eq:VnT \g_pyst_font_tl { minion } { \PystMiniontrue }
```

Esta única línea, en `core/pre/pyst-build.ltx`, es la bifurcación completa entre las dos partes de esta nota. `\str_if_eq:VnT` compara, carácter a carácter, el *valor* (V) de la variable con el texto literal (n) `minion`; si coinciden, activa `\ifPystMinion`, y es entonces cuando se sigue el camino de la *segunda parte* (Apartado 9). Para los otros 24 valores posibles, `\ifPystMinion` queda falso y se sigue el camino ordinario que ocupa el resto de esta primera parte.

## 3 pyst-fonts.ltx: el orquestador de la carga

`pyst-fonts.ltx` es el único fichero de la fase POST que decide *qué* perfil cargar. Todo perfil, una vez elegido, delega en `\PystDeclareFontProfile` (Apartado 4); pero decidir cuál, y qué hacer si el que se pidió no existe, es tarea de este fichero.

### 3.1 El registro de nombres, por si no se carga ningún perfil

Antes de tocar ningún perfil, `pyst-fonts.ltx` garantiza que tres comandos existan siempre, aunque no llegue a cargarse ninguno:

```
\providecommand\PystFontMain{(desconocida)}
\providecommand\PystFontSans{(desconocida)}
```

```
\providecommand\PystFontMono{(desconocida)}
```

`\PystFontMain`, `\PystFontSans` y `\PystFontMono` son los nombres legibles que el informe de depuración (`debug=true`) muestra al usuario. Con `\providecommand` y no `\newcommand`, cualquier perfil que se cargue después los sobrescribe sin avisar de nada; si ninguno llega a cargarse, caso límite, pero posible, el informe dice «(desconocida)» en lugar de fallar con un comando indefinido.

### 3.2 El declarador se carga antes que cualquier perfil

La línea

```
\pyst_input_twice:n { defs/pyst-fontprofile.def }
```

tiene que ir *antes* de intentar cargar ningún `pyst-font-<nombre>.def`, porque esos 24 ficheros no son más que una llamada a `\PystDeclareFontProfile`, y ese comando tiene que existir ya. Se carga con `\pyst_input_twice:n`, el mismo envoltorio

```
\ttcolorc{ExplSyntaxOff}  
\ttcolorc{input}  
\ttcolorc{ExplSyntaxOn}
```

que se describe en la nota de la fase PRE, aunque el propio fichero declare `\ProvidesExplFile`: el declarador está escrito en `expl3`, pero cada perfil, quien lo *llama*, está en régimen 2e, porque los nombres de fuente («TeX Gyre Termes», con espacios) tienen que sobrevivir a la tokenización, y en catcodes `expl3` un espacio desaparece.

### 3.3 El catálogo de fuentes válidas, construido por barrido

Antes de decidir si el perfil pedido existe, el sistema necesita saber *qué* perfiles existen, para poder nombrarlos en un mensaje de error si hace falta. Este catálogo no se mantiene a mano en ningún sitio: se construye barriendo el directorio `defs/fonts/` desde Lua.

```
function pyst_register_fonts()  
  local dir  
  local cls = kpse.find_file('pyst-pub.cls', 'tex')  
  if cls then  
    dir = (cls:match('^(.*/)') or './') .. 'defs/fonts/'  
    if not lfs.isdir(dir) then dir = nil end  
  end  
  if not dir then  
    local ref = kpse.find_file('defs/fonts/pyst-font-stixtwo.def', 'tex')  
    if ref then dir = ref:match('^(.*/)') end  
  end  
  if not dir then  
    return '(no se ha podido localizar defs/fonts/)'  
  end  
end
```

```

end
local t = {}
for f in lfs.dir(dir) do
  local k = f:match('^pyst%-font%-(.+)%.def$')
  if k then t[#t+1] = k end
end
table.sort(t)
if #t == 0 then
  return '(defs/fonts/ no contiene ningun perfil)'
end
return table.concat(t, ', ')
end

```

La función tiene *dos* anclas para localizar el directorio, y ninguna es arbitraria. La primera es `pyst-pub.cls`: si no se encuentra, no hay sistema que ejecutar, así que es una referencia segura. La segunda, de reserva, es precisamente el perfil `stixtwo`: no es un perfil cualquiera, es el que este mismo fichero carga como último recurso ([Apartado 3.4](#)), así que el sistema ya depende de su presencia en otro sitio; si desapareciera, el repliegue estaría roto de todas formas y hay problemas mayores que esta lista.

Esto no siempre fue así. Hasta la versión 2.1 había una única ancla, un perfil concreto elegido sin ningún criterio particular (`pagella`). Si algún día se renombraba o se retiraba ese perfil, la función devolvía '?' en silencio, y los dos mensajes de error sobre fuentes, los que existen precisamente para explicar un fallo, pasaban a decirle al usuario que las fuentes disponibles son '?', justo en el momento en que más necesita la lista completa. Con dos anclas independientes, ese modo de fallo deja de ser posible sin que además esté roto el propio repliegue.

El resultado se guarda, una sola vez por compilación, en una variable `expl3`:

```

\newcommand{\PystValidFonts}{\directlua{tex.sprint(
  pyst_register_fonts())}}
%...
\tl_gset:Nx \g_pyst_valid_fonts_tl { \PystValidFonts }

```

### 3.4 Decidir primero, cargar después

Con el catálogo ya construido, la decisión propiamente dicha es corta. Son tres situaciones, un aviso como mucho y un solo punto de carga:

```

\cs_new_protected:Npn \pyst_fonts_input_profile:n #1
{ \pyst_input_twoe:n { defs/fonts/pyst-font-#1.def } }
\cs_generate_variant:Nn \pyst_fonts_input_profile:n { V }

\tl_if_empty:NTF \g_pyst_font_tl
{
  \msg_warning:nnx { pyst } { font-fallback }
  { \tl_use:N \g_pyst_valid_fonts_tl }
  \tl_gset:Nn \g_pyst_font_tl { stixtwo }
}

```

```

}
{
  \file_if_exist:nF { defs/fonts/pyst-font-\tl_use:N
  \g_pyst_font_tl.def }
  {
    \msg_warning:nxxx { pyst } { font-not-found }
    { \tl_use:N \g_pyst_font_tl }
    { \tl_use:N \g_pyst_valid_fonts_tl }
    \tl_gset:Nn \g_pyst_font_tl { stixtwo }
  }
}

\pyst_fonts_input_profile:V \g_pyst_font_tl

```

`\file_if_exist:nF` es la comprobación real: no pregunta si el *nombre* está en el catálogo de [Apartado 3.3](#), esa variable es sólo para el mensaje, pregunta si el *fichero* `defs/fonts/pyst-font-<nombre>.def` existe de verdad en disco. La carga, al final, usa `\pyst_fonts_input_profile:V`, la variante que recibe el *valor* (V) de `\g_pyst_font_tl` en lugar del nombre de la variable, y se ejecuta una sola vez porque a esa altura la variable vale, con seguridad, el nombre de un perfil que existe: el pedido, o `stixtwo`.

**Tabla 2:** Las tres entradas al bloque.

| Se declaró                         | Qué se carga         | Aviso                       |
|------------------------------------|----------------------|-----------------------------|
| <code>font=&lt;válido&gt;</code>   | Ese perfil           | Ninguno                     |
| <code>font= (ausente)</code>       | <code>stixtwo</code> | <code>font-fallback</code>  |
| <code>font=&lt;inválido&gt;</code> | <code>stixtwo</code> | <code>font-not-found</code> |

Es deliberado que estos dos avisos, `\msg_warning:nxxx` para `font-not-found`, que nombra la fuente pedida y el catálogo, y `\msg_warning:nxx` para `font-fallback`, que sólo nombra el catálogo, sean avisos y no `\msg_error`.

#### 4 El declarador: \PystDeclareFontProfile

Cada uno de los 24 ficheros `defs/fonts/pyst-font-<nombre>.def` consiste, en su totalidad, en una sola llamada a este comando. Generalizar a otras tipografías ordinarias es tan fácil como seguir el esquema mostrado en cada uno de ellos. Por ejemplo, el perfil `termes` completo es el siguiente.

```

\ProvidesFile{pyst-font-termes.def}[2026/09/12 v2.3 Fuente TeX
  Gyre Termes]

\PystDeclareFontProfile{termes}{
  main          = TeX Gyre Termes,
  sans          = texgyreheroscn-regular.otf,
  sans-options  = {Ligatures=TeX,

```

```

                                ItalicFont=texgyreheroscn-italic.otf,
                                BoldFont=texgyreheroscn-bold.otf,
                                BoldItalicFont=texgyreheroscn-
bolditalic.otf},
mono                          = TeX Gyre Cursor,
face-regular                  = texgyreheroscn-regular.otf,
face-italic                   = texgyreheroscn-italic.otf,
face-bold                     = texgyreheroscn-bold.otf,
face-bolditalic               = texgyreheroscn-bolditalic.otf,
math-font                     = texgyretermes-math.otf,
math-package                  = lum-termes,
name-main                     = {TeX Gyre Termes},
name-sans                     = {TeX Gyre Heros Condensed},
name-mono                     = {TeX Gyre Cursor},
}

```

Vemos que el perfil declara *las dos* claves matemáticas: la fuente que prefiere (`math-font`) y el paquete al que replegarse si esa fuente no está (`math-package`). Los 24 perfiles ordinarios hacen lo mismo, y de ahí que las rutas A, B y C de [Apartado 4.4](#) sean las tres alcanzables para cualquiera de ellos. Observad también `sans-options`: cuando la sans se nombra por fichero hay que enumerar sus cuatro caras, porque nombrar un fichero concreto desactiva la búsqueda automática de la familia.

Por último. El fichero está encabezado por `\ProvidesFile`, no por `\ProvidesExplFile`: es la misma razón de catcodes de [Apartado 3](#), aplicada ahora al propio perfil. Y nótese que el perfil no llama a `\setmainfont` ni a `\setmathfont` en ningún sitio: se limita a declarar seis a ocho decisiones mediante *keyval*. Toda la ceremonia, qué llamar, en qué orden, con qué opciones, vive una sola vez en el declarador.

#### 4.1 Anatomía de la llamada: las claves

`\PystDeclareFontProfile` recibe dos argumentos obligatorios, el nombre del perfil y su *keyval*, y se declara sobre el espacio de claves propio `pyst / fontprofile`:

```

\NewDocumentCommand \PystDeclareFontProfile { m m }
{
  \tl_set:Nn \l_pyst_fp_key_tl { #1 }
  %...limpieza de variables locales...
  \keys_set:nn { pyst / fontprofile } { #2 }
  \pyst_fp_check:                                % validación previa
  \RequirePackage [no-math] {fontspec}           % paso 2
  \tl_use:N \l_pyst_fp_before_math_tl           % paso 3
  \pyst_fp_math_decide:                           % paso 4
  \pyst_fp_setup_text:                            % paso 5a
  \pyst_fp_math_emit:                             % paso 5b
  \pyst_fp_declare_faces:                         % paso 6
  \pyst_fp_register_names:
  \exp_args:NV \pyst_fp_set_lettrine:n \l_pyst_fp_lettrine_tl
  \tl_use:N \l_pyst_fp_extra_tl
}

```

La [Tabla 3](#) recoge las claves admitidas. Cualquier clave que no aparezca en esta tabla dispara `fontprofile-unknown-key`, un error que nombra la clave desconocida y el perfil en el que aparece, el mismo principio, de nuevo, que en la fase PRE: un perfil mal formado informa de su propio defecto en lugar de producir un «Undefined control sequence» cuatrocientas líneas después.

**Tabla 3:** Claves de `\PystDeclareFontProfile`.

| Clave                                                                                                                                                          | Significado                                                                                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>main, sans, mono</code>                                                                                                                                  | Las tres familias de texto. Cualquiera puede omitirse: esa llamada, sencillamente, no se hace.                                                                                                                                                 |
| <code>main-options,</code><br><code>sans-options,</code><br><code>mono-options</code>                                                                          | Opciones <code>fontspec</code> de cada familia. Con valor por defecto ( <code>Ligatures=TeX</code> las dos primeras, <code>Scale=MatchLowercase</code> la de <code>mono</code> ).                                                              |
| <code>face-regular,</code><br><code>face-italic,</code><br><code>face-bold,</code> <code>face-</code><br><code>bolditalic</code><br><code>face-titulari</code> | Las cuatro caras de las que salen las diez familias de visualización ( <a href="#">Apartado 4.8</a> ). Se declaran juntas o no se declara ninguna.                                                                                             |
| <code>math-font</code>                                                                                                                                         | Sustituye solo a la cara negrita-cursiva de <code>\titulari</code> . Solo la necesita <code>libertinus</code> , cuya sans no tiene negrita cursiva propia.                                                                                     |
| <code>math-options</code>                                                                                                                                      | Fuente matemática OpenType propia del perfil, <b>por fichero</b> , no por familia ( <a href="#">Apartado 5</a> ). Que esté declarada y que además esté instalada es lo que decide entre las rutas A, B/C y N ( <a href="#">Apartado 4.4</a> ). |
| <code>math</code> / <code>math-</code><br><code>package</code>                                                                                                 | Opciones de <code>\setmathfont</code> para la ruta A. Vacía por defecto, y a propósito: ninguno de los 24 perfiles la declara, porque una fuente matemática emparejada con su texto no necesita reescalar (a <a href="#">Apartado 4.7</a> ).   |
| <code>before-math</code><br><code>lettrine</code>                                                                                                              | Nombre del paquete <code>lum-&lt;perfil&gt;</code> , para la ruta B. Los dos nombres son la misma clave: <code>math</code> es la forma de la versión 2.2, conservada.                                                                          |
| <code>text-setup</code>                                                                                                                                        | Código arbitrario entre <code>fontspec</code> y la decisión matemática.                                                                                                                                                                        |
| <code>name-main,</code> <code>name-</code><br><code>sans,</code> <code>name-mono</code><br><code>extra</code>                                                  | Cuerpo de <code>\LettrineFontHook</code> , el que usa <code>\initial</code> . Por defecto, <code>\rmfamily\bfseries</code> .                                                                                                                   |
|                                                                                                                                                                | <code>auto</code> (por defecto) o <code>manual</code> , para omitir por completo el bloque de familias de texto.                                                                                                                               |
|                                                                                                                                                                | Nombres legibles, obligatorias las tres. Alimentan <code>\PystFontMain</code> y análogas.                                                                                                                                                      |
|                                                                                                                                                                | Código arbitrario, ejecutado al final de la llamada.                                                                                                                                                                                           |

#### 4.2 La validación previa: `\pyst_fp_check`:

Antes de cargar nada, el declarador comprueba que el perfil esté bien formado:

```

\int_compare:nNtT { \clist_count:N \l_pyst_fp_missing_clist } >
{ 0 }
{
  \int_compare:nNf { \clist_count:N \l_pyst_fp_missing_clist }
  = { 4 }
  {
    \msg_error:nnxx { pyst } { fontprofile-incomplete-faces }

    { \l_pyst_fp_key_tl }
    { \clist_use:Nn \l_pyst_fp_missing_clist {, ~ } }
  }
}

```

La lógica distingue dos situaciones que se parecen mucho pero significan cosas opuestas. **Cuatro** `face-*` ausentes es un perfil legítimo que no declara familias de visualización en absoluto, se repliega a los `\providecommand` genéricos de `pyst-fonts.ltx` §4,; entre **una y tres** ausentes es siempre un descuido, porque deja al menos un `\titulartocb` silenciosamente indefinido, y por eso dispara error nombrando exactamente cuáles faltan. La misma comprobación se repite para las tres claves `name-*`, obligatorias sin excepción porque alimentan el informe de depuración.

### 4.3 fontspec siempre primero

```
\RequirePackage [no-math] {fontspec}
```

Se centraliza aquí, con la opción `no-math` siempre puesta, y no en cada perfil por su cuenta como ocurría antes de la versión 2.2. La opción importa: sin ella, `fontspec` inicializaría por su cuenta un sistema matemático que el paso siguiente tiene que decidir él mismo.

### 4.4 Paso 4, decidir la ruta matemática

Este es el paso que cambia en la versión 2.3, y el centro de esta nota. Antes de esta versión, un perfil con matemáticas activas cargaba, sin más comprobación, el paquete `lum-<perfil>` (`lua-unicode-math`). Desde 2.3, la decisión pasa por `\pyst_fp_math_decide:`, que compara lo que el perfil declaró en `math-font` contra lo que de verdad hay instalado en la máquina donde se compila.

La función se divide en dos. La primera mitad no mira las fuentes de nadie: mira si el documento compone matemáticas.

```

\cs_new_protected:Npn \pyst_fp_math_decide:
{
  \tl_gclear:N \g_pyst_fp_mathroute_tl
  \tl_gclear:N \g_pyst_fp_mathused_tl

  \bool_if:NTF \g_pyst_module_math_bool

```

```

{ \pyst_fp_math_route: }
{ \tl_gset:Nn \g_pyst_fp_mathroute_tl { X } }
}

```

Con `math=false` no hay nada que decidir: no se carga `unicode-math`, no se carga ningún `lum-*` y no se fija ninguna fuente matemática. Es la **ruta X**, y es silenciosa porque no significa que falte nada: lo ha pedido el usuario. Este `\bool_if:NTF` es la comprobación que, hasta la versión 2.2, iba dentro de la carga incondicional del paquete `lum-*`; al sustituir esa carga por el enrutado hubo que traérsela aquí, porque sin ella todo documento con `math=false` habría empezado a cargar `unicode-math` por la ruta A.

Si el módulo de matemáticas está activo, la decisión propiamente dicha la toma la segunda mitad:

```

\cs_new_protected:Npn \pyst_fp_math_route:
{
  \tl_if_empty:NTF \l_pyst_fp_mathfont_tl
  {
    \tl_if_empty:NTF \l_pyst_fp_math_tl
    { \tl_gset:Nn \g_pyst_fp_mathroute_tl { N } }
    {
      \tl_gset:Nn \g_pyst_fp_mathroute_tl { B }
      \tl_gset_eq:NN \g_pyst_fp_mathused_tl
\l_pyst_fp_math_tl
      \pyst_fp_require:V \l_pyst_fp_math_tl
    }
  }
  {
    \pyst_fp_iffont:VTF \l_pyst_fp_mathfont_tl
    {
      \tl_gset:Nn \g_pyst_fp_mathroute_tl { A }
      \tl_gset_eq:NN \g_pyst_fp_mathused_tl
\l_pyst_fp_mathfont_tl
      \RequirePackage { unicode-math }
      \msg_info:nxxx { pyst } { math-unicode }
      { \l_pyst_fp_key_tl } { \l_pyst_fp_mathfont_tl }
    }
    {
      \tl_if_empty:NTF \l_pyst_fp_math_tl
      { \pyst_fp_math_route_c: }
      {
        \pyst_fp_iffile:VTF \l_pyst_fp_math_tl
        {
          \tl_gset:Nn \g_pyst_fp_mathroute_tl { B }
          \tl_gset_eq:NN \g_pyst_fp_mathused_tl
\l_pyst_fp_math_tl
          \pyst_fp_require:V \l_pyst_fp_math_tl
          \msg_warning:nxxxx { pyst } { math-lum-
fallback }

```

```

        { \l_pyst_fp_key_tl } {
\l_pyst_fp_mathfont_tl }
        { \l_pyst_fp_math_tl }
        }
        { \pyst_fp_math_route_c: }
        }
    }
}

```

Reducido a su esquema, son las cinco rutas de la [Tabla 4](#). El resultado se guarda en dos variables globales: `\g_pyst_fp_mathroute_tl`, con la letra de la ruta, que es lo que el paso 5b consulta para saber qué hacer, y `\g_pyst_fp_mathused_tl`, con *qué* se acabó cargando, el fichero de fuente en A y C, el nombre del paquete en B, que no la usa el mecanismo sino el informe de depuración ([Apartado 7](#)).

**Tabla 4:** Las cinco rutas de `\pyst_fp_math_decide:.`

| Ruta | Condición                                                                                                                      | Qué hace                                                                                    | Mensaje     |
|------|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|-------------|
| A    | <code>math-font</code> declarada, e <code>\IfFontExistsTF</code> confirma que <code>fontspec</code> puede resolverla de verdad | Carga <code>unicode-math</code> y usará esa fuente en el paso 5b                            | informativo |
| B    | <code>math-font</code> declarada pero ausente, y sí hay un <code>math/math-package</code> cuyo <code>.sty</code> existe        | Carga ese paquete <code>lum-&lt;perfil&gt;</code> ; las matemáticas quedan resueltas por él | aviso       |
| C    | <code>math-font</code> declarada pero ausente, y sin <code>lum-*</code>                                                        | Carga <code>unicode-math</code> ; el paso 5b usará STIX Two Math de repliegue               | aviso       |
| N    | Ni <code>math-font</code> ni <code>math-package</code>                                                                         | Nada: el perfil se gestiona sus propias matemáticas. Es el caso de <code>minion</code>      | ninguno     |
| X    | <code>math=false</code>                                                                                                        | Nada: el documento no compone matemáticas                                                   | ninguno     |

La distinción entre mensaje *informativo* y de *aviso* no es cosmética. La ruta A es la normal desde 2.3 en cualquier máquina con las 24 fuentes matemáticas instaladas, y así lo dice su mensaje (`math-unicode`, vía `\msg_info:nnxx`: «no hay nada que hacer»). Las rutas B y C, en cambio, significan que a esa instalación concreta le falta la fuente matemática que el perfil prefiere, y por eso se emiten con `\msg_warning:nnxxx`: la compilación no se detiene, pero el resultado no es el que el perfil declara como preferido.

N y X son las dos rutas silenciosas, y por la misma razón: ninguna de las dos significa que falte nada. En N lo ha declarado el perfil; en X, el usuario. Distinguir N de C es importante y no es una sutileza: una versión temprana de este enrutado las confundía, con el resultado de que un documento con `font=minion` cargaba `MinionPro` en la sección 1 de su perfil y, acto seguido, el declarador le superponía `unicode-math` y STIX Two Math. Compilaba sin un solo error y componía las fórmulas con la fuente equivocada.

Hay además una forma de la ruta B que no es un repliegue: un perfil que declare `math-package` y `no math-font` está pidiendo su `lum-*` de forma explícita, y por eso esa rama no emite mensaje. Ninguno de los 24 perfiles la usa hoy, todos declaran las dos claves, pero la forma sigue admitida.

Las dos variables globales se limpian al entrar, antes de decidir nada. Podría parecer superfluo, porque `pyst-fonts.ltx` garantiza que sólo se carga un perfil por documento, pero son globales y por tanto sobreviven a una declaración: sin ese borrado, las rutas N y X, que no asignan nada, quedarían vacías por suerte y no por construcción.

#### 4.5 Dos condicionales anidados

La rama que distingue B de C pregunta dos cosas: si el perfil declaró `math-package`, y si el `.sty` correspondiente existe en el árbol, con el comando `\pyst_fp_iffile:n` declarado como *predicado protegido* y en forma TF,

```
\prg_new_protected_conditional:Npnn \pyst_fp_iffile:n #1 { TF }
{ \file_if_exist:nTF { #1.sty } { \prg_return_true: } {
  \prg_return_false: } }
\cs_generate_variant:Nn \pyst_fp_iffile:nTF { VTF }
```

y las dos preguntas se anidan en lugar de conjugarse, de modo que cada una se evalúa en contexto de *ejecución*, que es donde `\file_if_exist:nTF` sí está definido para funcionar.

El orden correcto, que es el que ejecuta hoy el declarador, es:

```
\RequirePackage [no-math] {fontspec}      % paso 2
\tl_use:N \l_pyst_fp_before_math_tl      % paso 3
\pyst_fp_math_decide:                     % paso 4 , decide y
  carga el paquete
\pyst_fp_setup_text:                      % paso 5a, las familias
  de texto
\pyst_fp_math_emit:                       % paso 5b, el
  \setmathfont
\pyst_fp_declare_faces:                   % paso 6
```

La separación entre el paso 4 y el paso 5b, decidir y cargar el paquete por un lado, emitir `\setmathfont` por otro, con las familias de texto en medio, no es un capricho de organización: es la razón de [Apartado 4.7](#).

#### 4.6 `\setup_text::` las tres familias de texto

Antes de tocar las matemáticas, se establecen las tres familias de texto, cada una *como mucho una vez*

```
\cs_new_protected:Npn \pyst_fp_setup_text:
{
  \str_if_eq:VnT \l_pyst_fp_setup_tl { auto }
  {
    \defaultfontfeatures{Ligatures=TeX}
    \tl_if_empty:NF \l_pyst_fp_main_tl
    { \pyst_fp_setmain:VV \l_pyst_fp_main_opts_tl
      \l_pyst_fp_main_tl }
    \tl_if_empty:NF \l_pyst_fp_sans_tl
    { \pyst_fp_setsans:VV \l_pyst_fp_sans_opts_tl
      \l_pyst_fp_sans_tl }
    \tl_if_empty:NF \l_pyst_fp_mono_tl
    { \pyst_fp_setmono:VV \l_pyst_fp_mono_opts_tl
      \l_pyst_fp_mono_tl }
  }
}
```

Cada una de las tres llamadas está protegida por su propio `\tl_if_empty:NF`: omitir `sans` en el perfil, por ejemplo, significa que `\setsansfont` sencillamente no se invoca, en lugar de invocarse con un argumento vacío. Y cada asignación sale de una única línea del *keyval* del perfil, así que no puede repetirse por accidente: el defecto de `fira` es, por construcción, irreplicable.

#### 4.7 Paso 5b, el `\setmathfont`, después de `\setmainfont`

```
\cs_new_protected:Npn \pyst_fp_math_emit:
{
  \str_case:Vn \g_pyst_fp_mathroute_tl
  {
    { A } { \pyst_fp_setmath:VV \l_pyst_fp_mathopts_tl
      \l_pyst_fp_mathfont_tl }
    { C } { \pyst_fp_setmath:nn { Scale=MatchLowercase }
      { STIXTwoMath-Regular.otf } }
  }
}
```

Sólo las rutas A y C tienen rama en este `\str_case:Vn`. En la B no la necesitan, porque el `\setmathfont` lo ha hecho ya el paquete `lum-*`; en la N y la X no hay nada que emitir. Un `\str_case:Vn` sin rama coincidente no hace nada, así que las tres se resuelven solas sin una condición aparte.

**\str\_case:Vn y no \tl\_case:Nn** Parece que lo idiomático sería `\tl_case:Nn`, porque la ruta se guarda en una *token list*. No lo es, y la diferencia es de las que no se ven leyendo: `\tl_case:Nn` compara la variable contra **otras variables**, no contra contenido literal. Escrito como

```
\tl_case:Nn \g_pyst_fp_mathroute_tl { {A}{...} {C}{...} }
```

los casos `{A}` y `{C}` no son la letra «A» y la letra «C», sino dos secuencias de control que habría que haber declarado, de modo que **no casa nunca**. Se probó: con la ruta en A, ninguna rama se ejecutaba y no se emitía ningún `\setmathfont`. Y el documento *compilaba*, porque ni `\str_case` ni `\tl_case` consideran un error que no case ninguna rama.

Es exactamente el mismo modo de fallo que el enrutado desconectado de ?? y que la ruta N convertida en C: composición silenciosamente equivocada, cero avisos, cero errores. Por eso queda escrito aquí y en un comentario del propio fichero.

**Por qué la ruta A no lleva `Scale=MatchLowercase` y la C sí.** Mirando el listado salta a la vista que las dos ramas no son simétricas: la C pasa `Scale=MatchLowercase` escrito a mano, y la A pasa `math-options`, que está *vacía* por defecto y que ninguno de los 24 perfiles declara. La asimetría es deliberada.

Una fuente matemática emparejada con su tipografía de texto, Pagella con `texgyrepagella-math.otf`, Erewhon con `Erewhon-Math.otf`, ya viene dimensionada para ella por quien la diseñó. Aplicarle `Scale=MatchLowercase` la reescalaría contra una altura de equis que ya era la suya: no corrige nada y puede estropear el emparejamiento. En la ruta C, en cambio, STIX Two Math es una fuente *ajena* a la del texto, es el repliegue general del sistema, no la pareja de nadie, y ahí acomodarla sí es lo que hay que hacer.

De modo que `math-options` existe para el perfil cuya fuente matemática sí necesite un ajuste, no como hueco donde poner un valor por defecto que hoy nadie quiere. Por la misma razón no lleva `\initial:n`, al contrario que `main-options` y sus hermanas: el declarador hace `\tl_clear:N` sobre esa variable al empezar cada perfil, así que un valor inicial no llegaría a verse nunca, tendría que ponerse en el bloque de limpieza.

Este paso, el que de verdad emite `\setmathfont`, se llama *después* de `\pyst_fp_setup_text:`, es decir, después de que `\setmainfont` ya se ha ejecutado, y no antes. El motivo es `Scale=MatchLowercase`, la opción con la que la ruta C ajusta STIX Two Math: esa opción compara la altura de la fuente matemática contra la fuente *principal ya establecida*, así que si se llamara antes de `\setmainfont` compararía contra lo que hubiera antes, posiblemente la fuente por defecto del motor, no contra la que el documento va a usar de verdad. Con `lum-<perfil>` (ruta B) este orden nunca importó, porque era el propio paquete `lum-*` quien decidía cuándo y cómo llamar a `\setmathfont`, sin que PYST-PUB tuviera que preocuparse por ello. En la ruta B, por eso, `\pyst_fp_math_emit:` no hace nada: las matemáticas ya están resueltas.

#### 4.8 Las familias de visualización

De las cuatro caras (*face-regular/italic/bold/bolditalic*) salen diez familias de visualización, con `\newfontface`, no `\newfontfamily`:

```
\cs_new_protected:Npn \pyst_fp_face:Nn #1#2 { \newfontface #1 {#
  2} }
%...
\pyst_fp_face:NV \helveticam \l_pyst_fp_face_rm_tl
\pyst_fp_face:NV \helveticai \l_pyst_fp_face_it_tl
\pyst_fp_face:NV \helveticab \l_pyst_fp_face_bf_tl
\pyst_fp_face:NV \helveticax \l_pyst_fp_face_bi_tl

\pyst_fp_face:NV \titular \l_pyst_fp_face_bf_tl
\pyst_fp_face:NV \titulart \l_pyst_fp_face_bf_tl
\pyst_fp_face:NV \titulari \l_pyst_fp_face_ti_tl

\pyst_fp_face:NV \titulartoc \l_pyst_fp_face_rm_tl
\pyst_fp_face:NV \titulartoci \l_pyst_fp_face_it_tl
\pyst_fp_face:NV \titulartocb \l_pyst_fp_face_bf_tl
```

`\newfontface` y no `\newfontfamily` es una elección deliberada: la familia resultante *no* participa del cambio de serie y forma de NFSS, así que dentro de `{\helveticab ...}` un `\textit` no hace nada. Pasar a `\newfontfamily` cambiaría eso en todo `docstyle` que combine una familia de visualización con un cambio de serie, y ese repaso visual de regresiones queda para una versión posterior.

Nótese también el caso de `\titulari`: por defecto toma la cara negrita-cursiva (*face-bolditalic*), salvo que el perfil declare `face-titulari`, que la sustituye. Solo *libertinus* lo necesita, porque su sans no tiene una negrita cursiva propia.

#### 4.9 El registro de nombres legibles

```
\cs_new_protected:Npn \pyst_fp_register_names:
{
  \cs_gset:Npx \PystFontMain { \exp_not:V
    \l_pyst_fp_name_main_tl }
  \cs_gset:Npx \PystFontSans { \exp_not:V
    \l_pyst_fp_name_sans_tl }
  \cs_gset:Npx \PystFontMono { \exp_not:V
    \l_pyst_fp_name_mono_tl }
}
```

El último paso sobrescribe los tres `\providecommand` de [Apartado 3](#) con los nombres legibles que el perfil declaró en `name-main`, `name-sans` y `name-mono`. `\exp_not:V` evita una segunda expansión indebida del valor al fijarlo con `\cs_gset:Npx`: el nombre queda grabado tal cual, no como una instrucción que se reevalúe cada vez que se use `\PystFontMain`.

## 5 El anclaje por nombre de fichero

Tanto `math-font` como las tres fuentes que `pyst-fonts.ltx` §5 carga siempre (Apartado 6) se nombran por **fichero**, `STIXTwoMath-Regular.otf`, y no por **familia**, «STIX Two Math». La diferencia parece un detalle de estilo y no lo es y permite garantizar que dos documentos, generados en máquinas distintas, pero con la misma instalación de `TeXLive`, contengan exactamente los mismos glifos en las mismas posiciones.

Nombrando el fichero en lugar de la familia, la resolución deja de pasar por el mecanismo de familias del sistema operativo y pasa por `kpathsea`, que solo mira los árboles `TEXMF`. El documento compone así de forma idéntica en cualquier instalación, con independencia de qué otras fuentes de sistema tenga instaladas quien lo compile. Es también la razón de que `\pyst_fp_iffont:VTF` (Apartado 4.4) use `\IfFontExistsTF` de `fontspec` sobre el nombre de fichero declarado en `math-font`: la pregunta que de verdad importa no es «¿existe una fuente que se llame así?», sino «¿existe *este fichero* en un árbol `TEXMF`?».

## 6 `pyst-fonts.ltx` §5: el repliegue fijo de símbolos

Una vez resuelto el perfil, por la ruta que sea, hay un último bloque que se ejecuta siempre que `\ifPystMath` sea cierto y el perfil no sea `Minion`, con independencia de cuál haya sido la fuente matemática elegida:

```
\ifPystMinion
\relax
\else
\ifPystMath
\ifdefined\setmathfont
\newcommand*\pyst@mathrange[3]{%
\IfFontExistsTF{#1}{\setmathfont[range={#2}]{#1}}{%
\PackageWarningNoLine{pyst}{%
Fuente #1 no encontrada.\MessageBreak
Los glifos que pyst-fonts.ltx S5 toma de ella (#3)
}\MessageBreak
se componen con la fuente matematica del perfil%
}%
}%
\pyst@mathrange{STIXTwoMath-Regular.otf}
{"02217, "1D4F5, "029EB, "02B27, "0002F, "025FB, "03F5}
{...}
\pyst@mathrange{Erewhon-Math.otf}
{"023A7, "023A8, "023A9, "023AA, "023AB, "023AC, "023AD, "0007B,
"0007D}
{...}
\pyst@mathrange{LibertinusMath-Regular.otf}{\mathbb}{...}
\else
\PackageWarningNoLine{pyst}{%
```

```

\string\setmathfont\space no esta definido...%
}%
\fi

\let\mathds\mathbb
\def\blacksquare{\ensuremath\QED}
\let\blablap\pi
\renewcommand*{\pi}{\symup\blablap}
\def\square{\ensuremath\mdwhtsquare}
\fi
\fi

```

Las dos guardas, `\IfFontExistsTF` sobre cada fuente y `\ifdefined` sobre el propio `\setmathfont` corrigen una asimetría que el enrutado de [Apartado 4.4](#) dejaba a la vista en cuanto se conectó. El enrutado se repliega con cuidado cuando la fuente matemática *del perfil* no está instalada: rutas B y C, las dos con aviso y ninguna fatal. Este bloque no se replegaba en absoluto, y nombra tres fuentes **fijas** que no tienen por qué estar instaladas aunque la del perfil sí lo esté. Como un `\setmathfont` sobre una fuente ausente es un error fatal de `fontspec` y no un aviso, a una instalación a la que le faltara `erewhon-math` no le compilaba *ningún* documento con `math=true`, fuese cual fuese su `font=`. Ahora cada una avisa una vez y sigue; lo que se pierde es un puñado de glifos, que se componen con la fuente matemática del perfil.

La guarda sobre `\setmathfont` cubre la ruta B: allí quien carga `unicode-math`, y con él `\setmathfont`, es el paquete `lum-*`, por debajo, y si alguno dejara de hacerlo el comando no existiría al llegar aquí. Se usa `\ifdefined` y no `\ifundefined` porque este último, al comprobarlo, define `\setmathfont` como `\relax`: quien lo probara después lo encontraría «definido».

Estas tres llamadas se *suman* a lo que haya decidido el paso 5b, no lo sustituyen: un documento con `font=asana`, por ejemplo, tiene su matemática principal en `Asana-Math.otf` y, además, estas tres fuentes cargadas encima para un puñado de glifos concretos que `unicode-math` necesita poder resolver siempre, con independencia del perfil. Los siete puntos de código de la primera línea son, en orden, `\ast` (U+2217), `\mbfscr1` (U+1D4F5), los dos rombos (U+29EB, U+2B27), la barra de fracción (U+002F), el cuadrado (U+25FB) y la epsilon lunar  $\epsilon$  (U+03F5).

## 7 Qué se ve desde fuera: el informe de depuración

Todo lo anterior ocurre en silencio salvo por los mensajes de las rutas B y C. Con `debug=true`, el sistema además lo declara, y conviene saber dónde mirar porque el informe tiene *dos* formas y no dicen lo mismo por defecto.

La primera es el informe visual que compone `\PystDebugReport`, el que aparece al final del PDF. En su bloque 3, justo debajo del perfil elegido, dos filas resumen todo el recorrido de esta nota:

|         |         |
|---------|---------|
| Fuente: | erewhon |
|---------|---------|

```
Ruta matematica: A (unicode-math) --- Erewhon-Math.otf
```

La segunda es el resumen que va al terminal y al `.log` en `begindocument/before`, con la misma información bajo el rótulo `Math route:`. Las dos salen de la misma función, `\pyst_debug_mathroute:`, precisamente para que no puedan discrepar, el mismo principio que hace que el registro de nombres ([Apartado 4](#)) y las familias cargadas salgan de las mismas claves.

Lo que se lee en esa fila es la letra de la ruta, su mecanismo y, tras el guion, el contenido de `\g_pyst_fp_mathused_tl`:

```
A (unicode-math) --- texgyrepagella-math.otf
B (lum-*) --- lum-erewhon
C (unicode-math, STIX Two) --- STIXTwoMath-Regular.otf
N (ad hoc)
X (math=false)
```

La letra sola dice qué decisión se tomó; lo que sigue al guion dice cuál fue el resultado, que es lo que de verdad hace falta para saber si una ruta B o C se debe a lo que uno cree. Las glosas no están traducidas a propósito, `unicode-math`, `lum-*` y `math=false` son nombres de paquete y de opción, y «ad hoc» es igual en las dos lenguas del sistema; lo único traducido es el rótulo de la fila.

Un detalle de implementación que merece mención porque es el tipo de cosa que se descubre tarde: `\pyst_debug_mathroute:` se declara con `\cs_new:Npn` y no con `\cs_new_protected:Npn`. El resumen del terminal pasa por `\iow_term:x`, y una función protegida sobreviviría a esa expansión sin expandirse: el `.log` imprimiría el nombre de la función en lugar de su valor.

## 8 Las 24 correspondencias perfil → fuente matemática

La [Tabla 5](#) recoge, para cada uno de los 24 perfiles no-Minion, la fuente declarada en su clave `math-font`. Es, junto con el propio código de las rutas A/B/C, el dato que permite comprobar más tarde si la elección de una fuente concreta sigue siendo la acertada.

**Tabla 5:** Correspondencia perfil  $\rightarrow$  `math-font`.

| Perfil                | <code>math-font</code>     |
|-----------------------|----------------------------|
| arsenal               | ArsenalMath-Sans.otf       |
| asana                 | Asana-Math.otf             |
| bonum                 | texgyrebonum-math.otf      |
| concrete              | Concrete-Math.otf          |
| dejavu                | texgyredejavu-math.otf     |
| erewhon               | Erewhon-Math.otf           |
| euler                 | Euler-Math.otf             |
| fira                  | FiraMath-Regular.otf       |
| garamond              | Garamond-Math.otf          |
| gfsneohellenic        | GFSNeohellenicMath.otf     |
| ibmplex               | IBMPlexMath-Regular.otf    |
| lete-sans             | LeteSansMath.otf           |
| libertinus            | LibertinusMath-Regular.otf |
| lmodern               | latinmodern-math.otf       |
| luciole               | Luciole-Math.otf           |
| newcomputermodern     | NewCMMath-Regular.otf      |
| newcomputermodernsans | NewCMSansMath-Regular.otf  |
| oldstandard           | OldStandard-Math.otf       |
| pagella               | texgyrepagella-math.otf    |
| schola                | texgyreschola-math.otf     |
| stixtwo               | STIXTwoMath-Regular.otf    |
| termes                | texgyretermes-math.otf     |
| xcharter              | XCharter-Math.otf          |
| xits                  | XITSMath-Regular.otf       |

## Parte II

### El caso `font=minion`

**M**INION PRO es la única tipografía comercial del catálogo, y la única de los 25 perfiles cuyo fichero, `defs/fonts/pyst-font-minion.def`, no se reduce a una llamada a `\PystDeclareFontProfile`. El propio fichero lo explica en su cabecera con cuatro razones, que son el hilo conductor de esta segunda parte: es la única comercial y puede faltar; no existe una Minion Pro Math, así que su repliegue matemático no pasa por `unicode-math` sino por `MnSymbol`; sus matemáticas se resuelven *antes* de `fontspec`, no después. Gradúa sus propias caras con tamaños ópticos y redefine `\theequation`, dos cosas que ningún otro perfil hace.

#### 9 Por qué Minion no es declarativo del todo

Meter las cuatro particularidades anteriores en claves nuevas del declarador habría exigido inventar claves que solo Minion usaría, exactamente el defecto que `\PystDeclareFontProfile` vino a corregir (Apartado 4). La solución adoptada es más simple: Minion toma del declarador únicamente la parte que de verdad comparte con los demás, las diez familias de visualización, el registro de nombres y `\LettrineFontHook`, con `text-setup=manual` para que no intente fijar `main/sans/mono` por su cuenta, y resuelve todo lo demás, matemáticas y familias de texto, con código `fontspec` corriente, escrito directamente en el fichero del perfil.

#### 10 Matemáticas antes de `fontspec`

La primera particularidad, y la que más se aparta del resto del catálogo, es el orden: aquí las matemáticas se resuelven *antes* de cargar `fontspec`, no después como en el paso 5b de los 24 perfiles ordinarios (Apartado 4.7). El motivo es que `MinionPro.sty`, el paquete de terceros que aporta el soporte matemático, reclama para sí varios de los símbolos de fuente clásicos de  $\text{\LaTeX}$  (`OMS`, `OML`) antes de que `fontspec` tenga ocasión de tocarlos.

```

\ifPystMath
\IfFileExists{MinionPro.sty}
{%
  \RequirePackage[fullfamily, opticals, lf, minionint, mnsy,
    onlymath, mathlf]{MinionPro}
  \renewcommand*{\epsilon}{\ensuremath{\textup{\usefont{OML
}{cmr}{m}{n}\symbol{15}}}}
  \DeclareSymbolFont{otrossym}{OMS}{cmsy}{m}{n}
  \DeclareMathSymbol*{\mathbin}{otrossym}{"03}
  \renewcommand*{\ast}{\ensuremath{*}}
  \newcommand{\@scalemnsymbolfamily}[1]{%
    \DeclareFontShape{U}{##1}{m}{n}{
      <-6>[1.2]    ##15
      %...siete tramos ópticos más, hasta <12->...
    }{}%
    %...la misma escala de siete tramos para la forma
    negrita...
  }
  \@scalemnsymbolfamily{MnSymbolA}
  \@scalemnsymbolfamily{MnSymbolB}
  \@scalemnsymbolfamily{MnSymbolC}
  \@scalemnsymbolfamily{MnSymbolD}
  \RequirePackage{dsfont}
  \let\mathbb\mathds
  \let\mathbfup\mathbf
  \let\mathup\textup
  \renewcommand*{\pi}{\uppi}
  \newcommand*{\mbfscrl}{\bm{\ell}}
  \def\mdlgblklozenge{\filledmedlozenge}
}%
{%
  \PackageWarningNoLine{pyst}{%
    MinionPro.sty not found...\MessageBreak
    Falling back to Libertinus Math...%
  }%
  \PystMinionfalse
  \RequirePackage{unicode-math}
  \RequirePackage{libertinus-otf}
}%
\fi

```

`\IfFileExists`, de  $\text{\LaTeX}2\epsilon$ , no de `expl3`, es la comprobación de la que depende toda la rama. Si `MinionPro.sty` está instalado, se carga con las opciones que activan sus tamaños ópticos, sus cifras *old-style* y su soporte de `MnSymbol`, y a continuación se redefine a mano un puñado de símbolos, `\epsilon`, `\ast`, `\mathbb` (vía `\mathds` de `dsfont`), `\pi`, `\mbfscrl`, `\mdlgblklozenge`, que `MinionPro.sty` no cubre por sí solo y que en los 24 perfiles ordinarios resuelve `unicode-math` directamente. El bloque `\@scalemnsymbolfamily` gradúa además, a mano y en siete tramos de tamaño, las cuatro familias `MnSymbolA--D`, los símbolos matemáticos «de dibujo» que `MnSymbol` aporta, tanto en su forma

normal como en negrita, porque esas fuentes no traen tamaños ópticos propios y sin este bloque se verían desproporcionadas frente al resto de la composición.

### 10.1 El repliegue, y un efecto secundario que conviene conocer

Si `MinionPro.sty` no está instalado, es un paquete de terceros, no forma parte de un TeX Live estándar, la rama `F` de `\IfFileExists` avisa del hecho con `\PackageWarningNoLine` y se repliega a `Libertinus Math`, visualmente próxima a Minion Pro por compartir un diseño humanista y un contraste de trazo parecido, vía `unicode-math`, el mismo paquete que usan los 24 perfiles ordinarios, aquí solo como repliegue.

Conviene detenerse en una línea de esa rama que es fácil pasar por alto: `\PystMinionfalse`. Cuando el repliegue se activa, el perfil **deja de considerarse a sí mismo Minion** a efectos del resto del sistema. Esto tiene una consecuencia real, no cosmética, en dos sitios que dependen de `\ifPystMinion` y que se cargan *después* de este fichero:

- `pyst-fonts.ltx` §5 (Apartado 6), que se salta por completo si `\ifPystMinion` es cierto. Con el repliegue activo, `\ifPystMinion` ya es falso, así que las tres fuentes fijas de §5, `STIXTwoMath-Regular.otf`, `Erewhon-Math.otf`, `LibertinusMath-Regular.otf`, **sí se cargan**, superpuestas a `Libertinus Math`.
- `math/pyst-advanced-math.ltx` (Apartado 14), que hace lo contrario: se salta un paquete *si* `\ifPystMinion` es cierto. Con el repliegue activo, ese paquete **sí se carga**.

En otras palabras: un `font=minion` sin `MinionPro.sty` instalado no se limita a componer con una fuente matemática distinta; cambia además qué otros dos ficheros del sistema actúan sobre ese mismo documento, porque los tres consultan la misma variable global y esta rama es la única que la modifica en tiempo de carga del perfil.

## 11 Familias de texto manuales, con tamaños ópticos

Resueltas las matemáticas, el perfil carga `fontspec` sin más, `\RequirePackage[no-math]{fontspec}`, igual que en el paso 2 de los perfiles ordinarios (Apartado 4), y a partir de ahí fija `main`, `sans` y `mono` a mano, con `\setmainfont`, `\setsansfont` y `\setmonofont` corrientes, en lugar de a través de las claves `main/sans/mono` del declarador.

```
\setmainfont[
  Renderer = Basic,
  Ligatures = TeX,
  NFSSFamily = MinionPro,
  Scale = 1.0,
  SizeFeatures = {
    {Size={-8.4}, Font=* Capt},
    {Size={8.4-13}, Font=*},
```

```

    {Size={13-19.9},Font=* Subh},
    {Size={19.9-}, Font=* Disp}
  },
  BoldFont = MinionPro-Bold,
  BoldFeatures = { SizeFeatures = { ...los mismos cuatro
    tramos... } },
  ItalicFont = MinionPro-It,
  ItalicFeatures = { SizeFeatures = { ...idem... } },
  BoldItalicFont = MinionPro-BoldIt,
  BoldItalicFeatures = { SizeFeatures = { ...idem... } },
  Numbers = {OldStyle, Proportional},
  Kerning = On
]{Minion Pro}

```

La misma declaración se repite, prácticamente idéntica, en un segundo bloque que fija `\defaultfont` con `\newfontfamily`, no `\setmainfont`, más un `\greekfont` con las caras griegas correspondientes. Cierran esta sección `\setsansfont` sobre `MyriadPro-SemiCn` (con `Scale=MatchLowercase`, igual que la clave `sans-options` por defecto del declarador), `\setmonofont` sobre `NewCMMono10-Regular`, la misma mono que usan varios perfiles ordinarios, y un `\newfontface\minionlf` con cifras `Lining` en lugar de `OldStyle`, que se usará en [Apartado 13](#).

## 12 Lo que Minion sí comparte con los demás perfiles

Con las familias de texto y las matemáticas ya resueltas a mano, el fichero llama por fin al declarador, igual que cualquier otro perfil, pero con `text-setup=manual` para que no intente repetir lo que las dos secciones anteriores ya han hecho:

```

\PystDeclareFontProfile{minion}{
  text-setup      = manual,
  face-regular    = MyriadPro-SemiCn,
  face-italic     = MyriadPro-SemiCnIt,
  face-bold      = MyriadPro-BoldSemiCn,
  face-bolditalic = MyriadPro-BoldSemiCnIt,
  lettrine       = {\fontspec{MinionPro-Bold}},
  name-main      = {Minion Pro},
  name-sans      = {Myriad Pro Semicondensed},
  name-mono      = {New Computer Modern Mono},
}

```

Esta llamada sí atraviesa `\pyst_fp_check`: ([Apartado 4](#)) y sí produce las diez familias de visualización descritas en [Apartado 4.8](#), a partir de las caras *semicondensadas* de Myriad Pro. Pero Minion añade, a continuación y fuera del declarador, una distinción que ningún otro perfil hace: los titulares no usan la semicondensada, sino la **condensada**:

|                                                             |                            |
|-------------------------------------------------------------|----------------------------|
| <code>\renewfontface\titular{MyriadPro-BoldCond}</code>     | <code>% Condensed</code>   |
| <code>Bold</code>                                           |                            |
| <code>\renewfontface\titulart{MyriadPro-BoldCond}</code>    | <code>% Condensed</code>   |
| <code>Bold (títulos)</code>                                 |                            |
| <code>\renewfontface\titulari{MyriadPro-BoldCondIt}</code>  | <code>% Condensed</code>   |
| <code>Bold Italic</code>                                    |                            |
| <code>\renewfontface\titulartoc{MyriadPro-Cond}</code>      | <code>% Condensed (</code> |
| <code>TOC)</code>                                           |                            |
| <code>\renewfontface\titulartoci{MyriadPro-CondIt}</code>   | <code>% Condensed</code>   |
| <code>Italic (TOC)</code>                                   |                            |
| <code>\renewfontface\titulartocb{MyriadPro-BoldCond}</code> | <code>% Condensed</code>   |
| <code>Bold (TOC)</code>                                     |                            |

`\renewfontface` y no `\newfontface` porque las seis familias ya existen, las acaba de crear `\PystDeclareFontProfile`, con la semicondensada, y aquí solo se les asigna una cara distinta. Es la única parte de todo el fichero en la que Minion *corrige* lo que el declarador acaba de hacer, en lugar de sustituirlo o de complementarlo.

### 13 Numeración de ecuaciones con cifras de altura uniforme

El último bloque del perfil no tiene paralelo en ningún otro:

```
\AtBeginDocument{%
  \renewcommand{\theequation}{%
    \ifPystCap
      {\minionlf\thechapter.\arabic{equation}}
    \else
      {\minionlf\arabic{equation}}
    \fi
  }%
}
```

`\AtBeginDocument` retrasa la redefinición hasta que el documento arranca, porque `\theequation` y `\ifPystCap`, si el docstyle usa capítulos, no están garantizados hasta ese punto. El resultado envuelve el número de ecuación con `\minionlf` (Apartado 11), la cara de cifras `Lining` en lugar de las `OldStyle` que usa el resto del texto: unas cifras de altura uniforme se leen mejor dentro de la caja de una referencia numerada entre paréntesis, (3.14), que unas cifras de altura variable pensadas para fluir dentro de una línea de prosa.

### 14 Los otros dos sitios que consultan \ifPystMinion

`\ifPystMinion` no es exclusivo de este fichero: otros dos puntos del sistema, cargados después, la consultan para no deshacer lo que Minion ya ha resuelto por su cuenta.

El primero ya se ha descrito: `pyst-fonts.ltx` §5 (Apartado 6) empieza literalmente con `\ifPystMinion \relax \else ...`, si el perfil

es Minion, el bloque entero no hace nada, porque las tres fuentes fijas que carga son de `unicode-math` y Minion no pasa por ahí (salvo en el repliegue de [Apartado 10.1](#), que las activa precisamente porque deja de considerarse Minion).

El segundo aparece en `math/pyst-advanced-math.ltx`, y es de naturaleza distinta: no es una cuestión de diseño sino de un defecto conocido de interacción entre paquetes, documentado en el propio código:

```
% NOTE: mathfixs uses \ifdefined internally to detect unicode-
% math.
% With font=minion (MinionPro + MnSymbol, no unicode-math), this
% check fails under TL2026 due to kernel changes in \ifdefined
% evaluation order after MinionPro redefines math families.
% Skipped for minion; all other font profiles are unaffected.
\ifPystMinion\else
\RequirePackage[frac, rfrac, vfrac]{mathfixs}
\fi
```

El paquete `mathfixs`, correcciones de espaciado en fracciones, usado por `advanced-math=true`, decide internamente su propio comportamiento comprobando si `unicode-math` está cargado. Esa comprobación deja de ser fiable bajo TeX Live 2026 en cuanto `MinionPro.sty` ha redefinido las familias matemáticas clásicas, así que el sistema evita el problema sin más: con Minion, `mathfixs` sencillamente no se carga.

#### 14.1 En qué se parece y en qué difiere del camino ordinario

La [Tabla 6](#) resume, punto por punto, la comparación entre las dos partes de esta nota.

**Tabla 6:** Camino ordinario frente a `font=minion`.

|                                                             | 24 perfiles ordinarios                                                                                            | <code>font=minion</code>                                                                              |
|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Fuente matemática                                           | <code>unicode-math</code> + <code>math-font</code> , con repliegue a <code>lum-*</code> o STIX Two                | MnSymbol vía <code>MinionPro.sty</code> , con repliegue a <code>unicode-math</code> + Libertinus Math |
| Orden                                                       | Matemáticas <i>después</i> de <code>fontspec</code> (paso 5b)                                                     | Matemáticas <i>antes</i> de <code>fontspec</code>                                                     |
| Familias de texto                                           | Vía las claves <code>main/sans/mono</code> del declarador ( <code>text-setup=auto</code> )                        | A mano, con <code>fontspec</code> corriente ( <code>text-setup=manual</code> )                        |
| Tamaños ópticos                                             | No se gestionan; cada perfil usa un único master                                                                  | Graduados a mano en cuatro tramos por forma                                                           |
| \$5 de <code>pyst-fonts.ltx</code><br><code>mathfixs</code> | Se ejecuta siempre, saltándose las fuentes que no estén instaladas<br>Se carga si <code>advanced-math=true</code> | Se salta por completo (salvo en repliegue)<br>Nunca se carga                                          |
| Declarador                                                  | Toda la ceremonia                                                                                                 | Solo caras, nombres y <code>\LettrineFontHook</code>                                                  |