
typst: ¿el reemplazo de L^AT_EX?

Una alternativa moderna y más amigable a L^AT_EX

Matías Fernández Taipe

10 de Mayo del 2023

- # Introducción
- typst** es un lenguaje de tipografía de código abierto para escribir documentos de alta calidad tipográfica.
 - Creado por desarrolladores insatisfechos con $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, ofrece una sintaxis sencilla y moderna.

typst

1

2

3

4

5

24

25

26

27

28

29

30

31

32

33

34

35

36


37

< Typst File Edit View Help

Ag B I U H ≡ ≡ Σ <> @

Johanna's Typst > Space Mail

🗨️ ✓ - 100% +

 Share ⬇️

```
#import "template.typ": *
#show: paper.with(
  title: [Towards Swifter Interstellar Mail Delivery],
  date: [May 17th, 2022],
  ...
)

= Introduction Egon
Our concept suggests three ways that A-Mail can be best utilized.

- First is to reduce the probability of the failure of a space mission. This
  problem is known as the Mars problem and suggests problems with human
  communication.

- As A-Mails are written using pen and paper, no digital technology is needed for
  short and long distance communication. This suggests a possibility of reducing
  the communication monopoly currently held by an entity known as the "internet".

- High round-trip times required for communication between Mars and Earth inhibits
  successful human developments on the planet. In contrast, the delivery speed of
  an A-Mail can be determined through this simple formula:
  $ v(t) = \lim_{t \rightarrow \infty} \int_0^t c \cdot \sqrt{t^2} \, dt $

#figure(
  image("a-mail.svg"),
```

Towards Swifter Interstellar Mail Delivery

Johanna Swift
Delivery Institute

Egon Stellaris
Space Institute

Oliver Liam
Mail Institute

May 17th, 2022

Until there is a definitive answer to the mystery of the dead star,
please use the old postal system to submit your question and report
the location of missing letters to the P.I.

ABSTRACT

Recent advances in space-based document processing have
enabled faster mail delivery between different planets of a
solar system. Given the time it takes for a message to be
transmitted from one planet to the next, its estimated that
even a one-way trip to a distant destination could take
up to one year. During these periods of interplanetary
mail delivery there is a slight possibility of mail being
lost in transit. This issue is considered so serious that
space management employs P.I. agents to track down and
retrieve lost mail. We propose A-Mail, a new anti-matter
based approach that can ensure that mail loss occur-
ring during interplanetary transit is unobservable and
therefore potentially undetectable. Going even further, we
extend A-Mail to predict problems and apply existing and
new best practices to ensure the mail is delivered without
any issues. We call this extension AF-Mail.

Reference: Johanna Swift, Egon Stellaris, Oliver Liam. Towards




Figure 1: Visualization of the FTL Earth-to-Mars
communication capabilities enabled by A-Mail.

- High round-trip times required for communi-
cation between Mars and Earth inhibits successful
human developments on the planet. In contrast,
the delivery speed of an A-Mail can be deter-
mined through this simple formula:

$$v(t) = \lim_{t \rightarrow \infty} \int_0^t c \cdot \sqrt{t^2} \, dt$$

Problemáticas que soluciona **typst**

- Compilación rápida a PDF
- Trabajo colaborativo en tiempo real
- Creación sencilla de macros y templates

Ventajas en comparación a \LaTeX

- Previsualización en tiempo real
- Mejores mensajes de errores
- Sintáxis más intuitiva de aprender

Comparativa \LaTeX vs **typst**

Fibonacci.tex

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{geometry}

\geometry{a4paper, margin=2cm}
```

```
\pagestyle{empty}
```

```
\begin{document}
```

```
\begin{center}
  {\LARGE \textbf{Secuencia de Fibonacci}}
\end{center}
```

```
\section{Definición Recursiva}
```

La secuencia de Fibonacci se define de forma recursiva como:

```
\begin{equation*}
  F_n = \begin{cases}
    0 & \text{si } n = 0 \\
    1 & \text{si } n = 1 \\
    F_{n-1} + F_{n-2} & \text{si } n > 1
  \end{cases}
\end{equation*}
```

```
\section{Ecuación Cerrada}
```

La secuencia de Fibonacci también se puede expresar mediante la siguiente ecuación cerrada:

```
\begin{equation*}
  F_n = \frac{1}{\sqrt{5}} \left[ \left( \frac{1 + \sqrt{5}}{2} \right)^n - \left( \frac{1 - \sqrt{5}}{2} \right)^n \right]
\end{equation*}

\end{document}
```

Fibonacci.typ

```
#set page(paper: "a5", margin: 2cm)
#set heading(numbering: "1 ")

#align(center)[
  #text(size: 18pt, weight: 600)[Secuencia de Fibonacci]
]
```

= Definición Recursiva

La secuencia de Fibonacci se define de forma recursiva como:

```
$ F_n = cases(
0 &"si" n = 0,
1 &"si" n = 1,
F_(n-1) + F_(n-2) &"si" n > 1
) $
```

= Ecuación Cerrada

La secuencia de Fibonacci también se puede expresar mediante la siguiente ecuación cerrada:

```
$ F_n = 1 / sqrt(5) [ ((1 + sqrt(5)) / (2))^n - ((1 - sqrt(5)) / (2))^n ] $
```

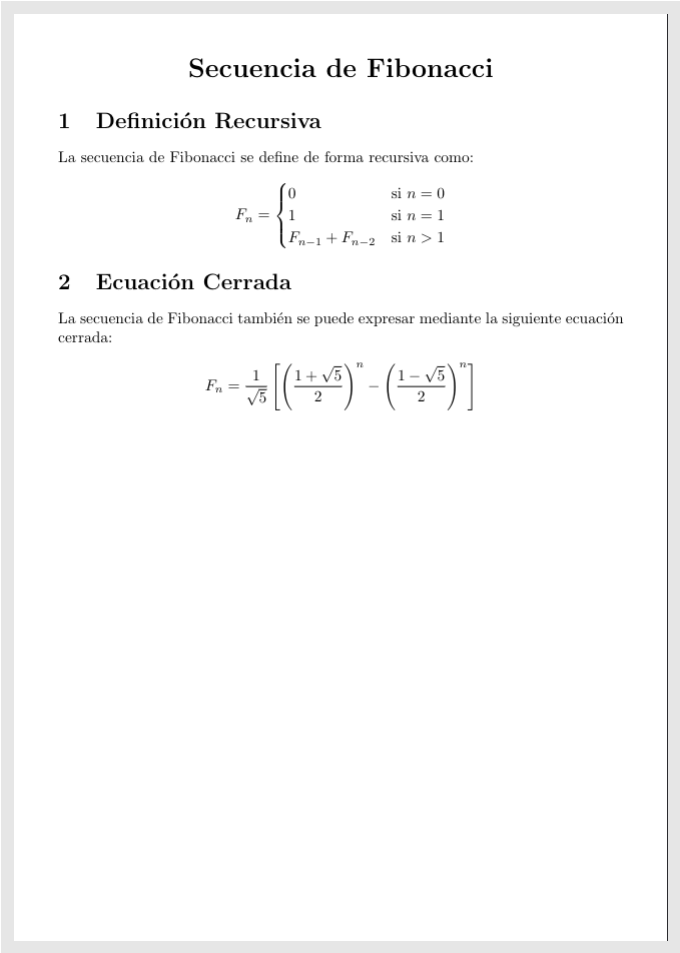


Figure 1: \LaTeX

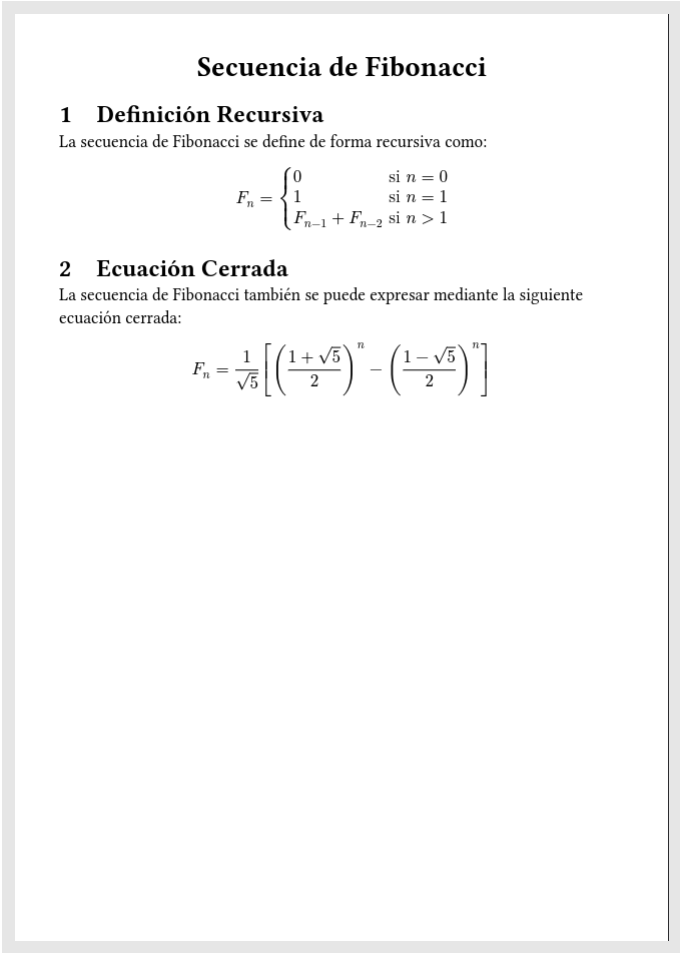


Figure 2: **typst**

Cómo ocupar **typst**

Aplicación Web

Se puede ocupar **typst** en línea desde la página oficial de **Typst** . Para ello, es necesario crear una cuenta de usuario y luego iniciar sesión en la página web. Una vez iniciada la sesión, se puede comenzar a utilizar **typst** en línea.

CLI

- Descarga el binario desde el repositorio de [GitHub](#) para tu sistema operativo.
- Usa los siguientes comandos desde la terminal:
 - `typst compile <nombre>`: compila el archivo de **typst** con el nombre especificado.
 - `typst watch <nombre>`: compila el archivo de **typst** con el nombre especificado y lo actualiza automáticamente cada vez que se guardan cambios en el archivo.

Nota: yo utilizo el comando `typst watch <nombre-ty>` y `zathura <nombre-pdf>` para ver los cambios en tiempo real.

Integración con VSCode

La extensión **typst LSP** permite resaltar sintaxis, reportar errores, autocompletar código y ayuda con la firma de funciones. Además, compila a PDF al guardar y puede configurarse para compilar en tiempo real o deshabilitar la función.

Documentación

La documentación de **typst** se encuentra en [Typst Docs](#) .

Modos de **typst**

typst tiene tres modos:

1. **markup**
2. **código**
3. **matemático**

Modo markup

Modo por defecto, y se utiliza para escribir texto normal, con algunas características que enriquecen el texto al estilo de Markdown.

Ejemplo markup

= Este es un título

Hola, este es un párrafo normal.

- item 1
- item 2
 - + subitem 1
 - + subitem 2

== Subtítulo

Letras en *_cursiva_* y ***negrita***.

Este es un título

Hola, este es un párrafo normal.

- item 1
- item 2
 1. subitem 1
 2. subitem 2

Subtítulo

Letras en *cursiva* y **negrita**.

Modo matemático

Para entrar en modo matemático se utiliza `$<ecuaciones>$` al igual que en Markdown o \LaTeX . Y si se quiere escribir una ecuación en una línea aparte, se utiliza `$ <ecuaciones> $`.

typst tiene una gran cantidad de símbolos matemáticos, y se pueden consultar en la [documentación](#).

Ejemplo modo matemático

= Ecuación en línea

El área de un círculo de radio r es $A = \pi r^2$.

= Ecuación en línea aparte

Llamaremos \mathcal{A} al conjunto definido por $\mathcal{A} = \{x \in \mathbb{R} \mid x > 0\}$

Ecuación en línea

El área de un círculo de radio r es $A = \pi r^2$.

Ecuación en línea aparte

Llamaremos \mathcal{A} al conjunto definido por

$$\mathcal{A} = \{x \in \mathbb{R} \mid x > 0\}$$

Modo código

Vamos a ver algunas funciones y macros que permiten darle formato al documento. De todos modos hay una lista más extensa de funcionalidades.

Normalmente sigue el patrón:

```
#función(argumentos) [
  contenido
]
```

El `#` se utiliza para desambiguar el modo código del modo markup, cuando ya se está en el modo código y se quiere ocupar de nuevo una función no es necesario volverla a ocupar.

Función `#text`

La función `#text` permite darle formato al texto, como cambiar el tamaño, el color, la alineación, etc.

```
#text(blue)[\Typst] es un lenguaje
de #text(style:"italic")[tipografía]
de código abierto para escribir
documentos de #text(font:"Ubuntu
Mono")[alta calidad tipográfica].
```

Typst es un lenguaje de *tipografía* de código abierto para escribir documentos de alta calidad tipográfica.

Función **#image**

```
Y con ustedes, la #text(blue)
[capybara] más famosa de #text(blue)
[\Typst]
#image("src/capybara.jpg", width:
50%)
```

Y con ustedes, la **capybara** más
famosa de **Typst**



Función `#link`

Puedes encontrar más información en
`#link("https://typst.app/")[\Typst]`

Puedes encontrar más información en
[Typst](https://typst.app/)

Función ``#lorem``

```
#lorem(20)
```

Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore
et dolore magnam aliquam quaerat.

Función #let

La función #let permite crear nuestras propias variables y funciones.

```
#let faboloso(term, color: blue) = {  
  text(color, box[||| #term |||])  
}
```

Tú eres #faboloso[guapisimo]!

Yo soy #faboloso(color: purple)
[faboloso]!

Tú eres ||| guapisimo |||!

Yo soy ||| faboloso |||!

Función `#import`

La función `#import` permite importar variables y funciones de otros archivos. Supongamos tenemos el archivo `faboloso.typ` con la función `faboloso`:

```
#let faboloso(term, color: blue) = {
  text(color, box[||| #term |||])
}
```

Luego lo podemos llamar de la siguiente manera:

```
#import "faboloso.typ": faboloso
```

```
Tú eres #faboloso[guapisimo]!
```

```
Yo soy #faboloso(color: purple)
[faboloso]!
```

Tú eres ||| guapisimo |||!

Yo soy ||| faboloso |||!

Función #set

La función #set permite establecer reglas que se le aplican a los elementos que se indiquen. Los elementos que se pueden modificar son todos aquellos que tienen asociado alguna función.

```
#set text(font: "Ubuntu Mono")  
#set text(fill: blue)  
#set par(justify: true)  
  
#lorem(20)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat.

Función `#set` y scope

Las reglas que se establecen con `#set` tienen el scope del bloque de contenido en donde se encuentran.

```
Esto se ve afectado #[  
  #set list(marker: [--])  
  - Dash  
]
```

Pero esto no:
- Bullet

Esto se ve afectado

– Dash

Pero esto no:

• Bullet

Función #show

La función show lo que hace es establecer una regla que va a reemplazar el elemento que se indique por lo que nosotros queramos.

```
#show "Ping": "Pong"  
Ping
```

Pong

Función #show obtener el valor

También se puede acceder al valor de la regla con la siguiente sintaxis:

```
#show "Hola mundo": val =>  
[#text(blue)[#val]]  
Hola mundo
```

Hola mundo

Función `#show` con selector general

Hay veces que queremos pasar el *resto* del documento como parámetro, para eso se utiliza el selector general.

```
#show: rest => columns(2, rest)
```

```
= Introduction
```

```
#lorem(15)
```

```
= Related Work
```

```
#lorem(2)
```

```
= Related Work
```

```
#lorem(2)
```

Introduction Related

Lorem ipsum dolor
sit amet,
consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore.

Work

Lorem ipsum.

Related

Work

Lorem ipsum.

Discusión final

Podrá **typst** reemplazar a \LaTeX ?

Probablemente **no** a corto plazo:

- Aún está en fase de desarrollo y por ende le quedan muchas funcionalidades y errores por corregir.
- \LaTeX es un estándar de facto en el mundo científico.

Pese a esto, **typst** es una herramienta muy interesante y con mucho potencial.

Links de interés

- [Página oficial](#)
- [Documentación oficial](#)
- [Repositorio de GitHub](#)
- [Awesome Typst](#)
- [Discord oficial](#)

Gracias por su atención!

