Sistema FIEB

SENAI CIMATEC

PELO FUTURO DA INOVAÇÃO

minicurso

LaTeX

Aziel Freitas

27 de maio de 2021

Manual

Minicurso LaTeX

Introdução e boas práticas em LaTeX

Aziel Freitas

Pesquisador bolsista

aziel.freitas@fbter.org.br

Centro de Competência em Robótica e Sistemas Autônomos

Sistema FIEB | SENAI CIMATEC

PELO FUTURO DA INOVAÇÃO

27 de maio de 2021

Sumário

Lista de Figuras

1. Hello LaTeX!

Esse manual irá sintetizar boas práticas no uso de LaTeX de forma prática e objetiva, sempre tentando incluir esclarecimentos adicionais que ajudem a fixar o conteúdo! São inúmeros os tipos de documentos que podem ser criados com LaTeX, porém vamos tratar especificamente da *classe* `report`. Inicialmente apresentaremos declarações mínimas para compor um documento e tentaremos cobrir alguns dos pacotes mais recentes ou que oferecem as funcionalidades mais comuns e interessantes.

Processadores de texto como Word, LibreOffice Writer e afins, utilizam o paradigma "WYSIWYG", que significa *What You See Is What You Get*. Neles, você, autor(a), concebe o seu documento ao passo que ele vai sendo exibido "instantaneamente", "ao vivo". Editores assim possuem uma interface amigável e todas as suas funcionalidades estão à mostra em menus e ícones.

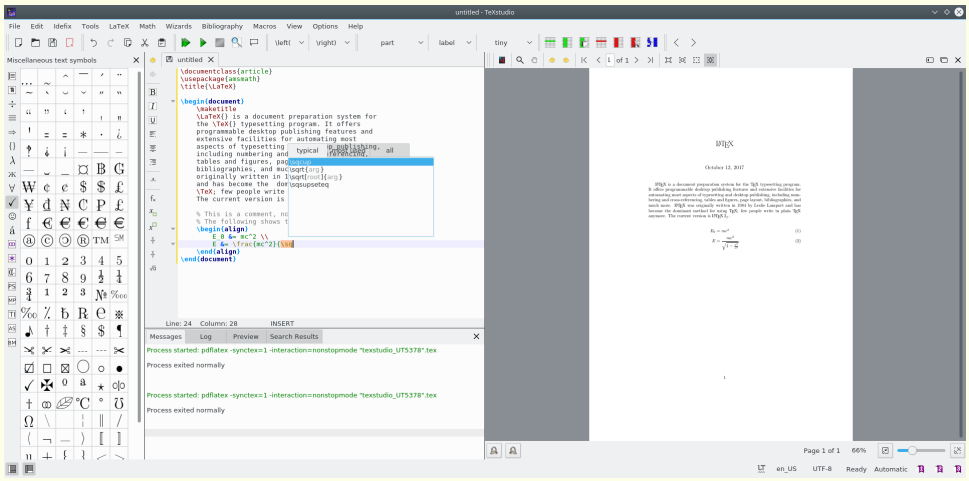


Figura 1.1: O TeXstudio tentando ser "amigável" como o Word.

O nosso processador de texto LaTeX, seus recursos não se mostram prontamente. O seu paradigma é "WYSIWYM", ou *What You See Is What You Mean*. Neste caso, um compilador terá que interpretar os comandos que foram dados e daí produzir um documento de texto legível de fato. Essa abordagem é mais laboriosa, especialmente no início, mas o usuário tem maior liberdade. Existem editores como o **TeXstudio** que tentam realizar uma mistura com "o melhor dos dois mundos", trazendo uma interface gráfica mais amigável com os recursos mais frequentemente utilizados. No fim das contas, é necessário que você saiba o que deseja ter no seu documento para então buscar como implementar no seu código .tex. Diante de possibilidades incontáveis, a escrita em código ainda é o meio mais viável para o máximo de customização!

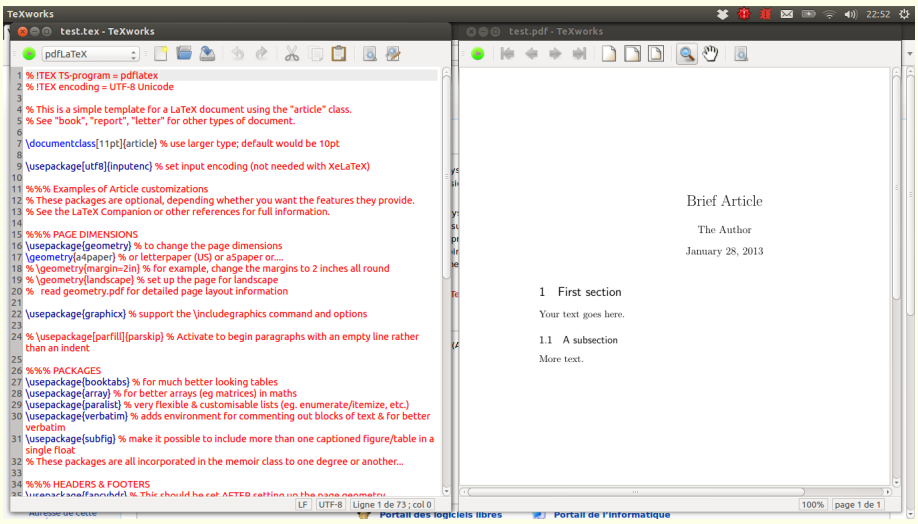


Figura 1.2: A interface *clean* do TeXworks.

1.1 Estruturação

Ajude o LaTeX ao organizar o seu documento segundo **alguma estrutura**! Estruturação é essencial no conteúdo, e também no seu código `.tex`. Modularizar o código é uma ótima ideia porque tanto você quanto o compilador podem se concentrar apenas no trecho que está sendo produzido. **Se o seu documento for grande**, com apêndices, anexos e figuras, **você vai gostar muito** que o compilador **não perca tempo** em trechos que não foram alterados!

> **!** Crie um arquivo para ser o **preâmbulo**; crie um arquivo principal que irá "chamar" cada capítulo/seção do seu documento.

1.2 Declarações - a barra invertida

Em LaTeX comandos são declarados através de palavras (tipicamente em inglês) precedidas por uma \. Muitas vezes, os comandos esperam que você informe **parâmetros**, que devem ir dentro de colchetes ou chaves. Vamos ver algumas declarações básicas em qualquer tipo de documento na ordem em que tipicamente aparecem. Não perca essas explicações!

`usepackage` O comando `usepackage` informa para o compilador que você deseja "invocar" as funcionalidades de algum *package*. Em LaTeX, esses *packages* são códigos que concedem funcionalidades para melhor criarmos nossos documentos. Este documento, por exemplo, utiliza o *package* `fvextra`[1], cuja função será explicada. Para adicionarmos a funcionalidade deste pacote devemos declarar, preferencialmente no início do código `.tex`: `\usepackage{fvextra}`. Pode-se dizer que a declaração `\usepackage{}` "pega emprestado" de outros arquivos `.cls`, `.sty`, e `.tex` linhas de código para implementar no nosso código alguma funcionalidade. O **CTAN** é um grande repositório com muitos *packages* e os seus manuais de uso.

`documentclass` É esperado que todo documento em LaTeX possua uma classe – `book`, `report`, `article`, `slides`, `letter`, etc. As classes são nomeadas por modelos de documento. Atrás das cortinas o que ocorre é: ao declarar uma classe, você estará "pegando emprestado" códigos de outros arquivos que permitem a confecção de um documento da classe escolhida. Se

[1] Geralmente os pacotes *fancy* são aprimoramentos de pacotes "tradicionais" ou funcionalidades já existentes no LaTeX básico. Este é um aprimoramento do `verbatim`, que significa "literalmente".

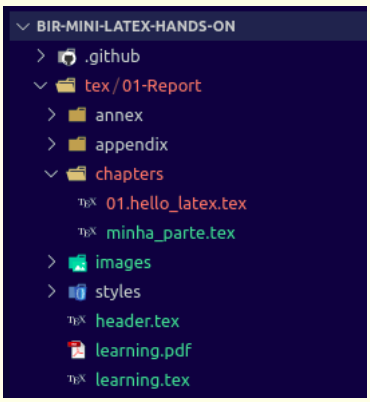


Figura 1.3: Pastas dedicadas a partes específicas do documento.

você for fazer um relatório, por exemplo declare `\documentclass[brazilian]{report}`. Note o parâmetro `brazilian`. Ele informa ao compilador que o documento está em português do Brasil, assim, os pacotes que usufruem da informação de idioma poderão funcionar adequadamente.

`begin` e `end`

Agora que você declarou a `\documentclass` e alguns *packages* que vão te auxiliar, é hora de informar que o corpo do documento vai começar a ser criado. Aí entra a declaração `\begin{document}`. Logicamente, toda declaração `\begin{}` deve trazer à tona **em algum momento** uma declaração `\end{}`. Os pares `\begin{}` ⇒ `\end{}` compõem os **ambientes** no LaTeX.

> **!** Tudo que vem após a declaração `\end{document}` será ignorado!

A estruturação comentada na Seção **1.1** pode ser feita pela separação do documento em vários arquivos `.tex`. Recomendamos a organização do seu código em pastas de uma forma parecida como na Fig. **1.3**. Nela você pode ver os arquivos `01.hello_latex.tex` e `minha_parte.tex`, dois arquivos de código. Como eles se relacionam? Vamos explicar abaixo.

`input`

Suponha que você está editando um documento com um colega. Ele envia para você o arquivo `minha_parte.tex`, e o seu conteúdo corresponde ao da caixa azul . Você deseja inserir o arquivo dele no arquivo principal, `01.hello_latex.tex` – é simples, declare `\input{./minha_parte}` no corpo do texto e a inserção é feita no exato ponto onde a declaração aparece. A declaração indica para o compilador "insira o arquivo `minha_parte.tex` que se encontra na mesma pasta do arquivo principal". O diagrama abaixo ilustra o efeito desta declaração.

Texto do seu amigo

As equações de Maxwell na forma diferencial são:

$$\nabla \times \vec{H}(r,t) = \vec{J}(r,t) + \frac{\partial \vec{D}(r,t)}{\partial t}$$
$$\nabla \times \vec{E}(r,t) = -\frac{\partial \vec{B}(r,t)}{\partial t}$$
$$\nabla \cdot \vec{D}(r,t) = \rho(r,t)$$
$$\nabla \cdot \vec{B}(r,t) = 0.$$

Texto no arquivo principal

Vamos então estudar as equações de Maxwell.

As equações de Maxwell na forma diferencial são:

$$\nabla \times \vec{H}(r,t) = \vec{J}(r,t) + \frac{\partial \vec{D}(r,t)}{\partial t}$$
$$\nabla \times \vec{E}(r,t) = -\frac{\partial \vec{B}(r,t)}{\partial t}$$
$$\nabla \cdot \vec{D}(r,t) = \rho(r,t)$$
$$\nabla \cdot \vec{B}(r,t) = 0.$$

Na primeira equação tem-se o operador rotacional aplicado ...

1.3 Indentação

Uma boa prática em quase qualquer linguagem de programação é a *indentação*, um ajuste na margem esquerda a fim de deixar legíveis subgrupos de comandos *aninhados*. Vamos exemplificar com o uso de um ambiente básico, `enumerate`. Veja o código abaixo e o seu resultado no lado direito:

Código 1.1

```
1 Uma␣listagem␣qualquer
2 \begin{enumerate}
3 ␣␣\setlist[enumerate]{label*={\arabic*.}}
4 ␣␣\item␣Esse␣é␣o␣primeiro␣item;
5 ␣␣\item␣O␣segundo␣item␣vem␣aqui;
6 ␣␣\begin{enumerate}
7 ␣␣␣␣\item␣Opa,␣um␣subitem!
8 ␣␣\end{enumerate}
9 \end{enumerate}
```

Resultado 1.1

Uma listagem qualquer

1. Esse é o primeiro item;
2. O segundo item vem aqui;
   2.1. Opa, um subitem!

A indentação já é algo familiar em listagens, no lado direito. Veja como o código ganha legibilidade quando usamos a indentação nele.

Usando packages

Falamos dos comandos fundamentais – as declarações mencionadas na Seção **1.2** aparecem em qualquer documento, e apresentamos `input`, uma declaração muito importante para ajudar a organizar o processo criativo do seu documento. Nos concentraremos num tipo específico de documento para daí nos nos voltarmos para funcionalidades típicas de documentos relacionados a engenharia, física e matemática. Vamos nos aprofundar um pouco mais no uso dos *packages*, abordando pacotes usados **nesse documento**!

Os pacotes possuem seus próprios manuais, contendo descrições sobre as funcionalidades e como empregá-las. Já no momento da declaração do pacote é possível declarar parâmetros entre colchetes. Estes terão efeito ao longo de todo o documento. Em alguns pacotes, estes efeitos podem ser alterados. Pacotes estão sempre "conversando" entre si e frequentemente existe relação de dependência entre eles. São constantemente atualizados e de certa forma há uma "seleção natural" que torna uns obsoletos em detrimento a outros. É preciso escolher cuidadosamente os pacotes declarados pois podem haver conflitos e informação pode ser encontrada nas suas documentações ou em fóruns on-line.

1.4 babel

`\usepackage[brazil]{babel}` invoca o pacote `babel` para o uso do idioma português no Brasil. Ele define regras de hifenização e outros aspectos regionais do documento, como a palavra "Capítulo" ao começo de cada capítulo (caso ela seja exibida), ou o formato da data no comando `\date`.

1.5 inputenc

Declarar `\usepackage[utf8]{inputenc}` permite a correta codificação dos caracteres especiais (acentuados) que utilizamos no português. Isso permite por exemplo, que os caracteres especiais do arquivo `.pdf` gerado pelo código LaTeX sejam copiados adequadamente.

1.6 geometry

É o pacote que define aspectos geométricos do documento. Margens, dimensionamentos, formato de papel (retrato ou paisagem). Para melhor aproveitar o espaço vertical da página, declaramos

```
\usepackage[a4paper,top=20mm,bottom=20mm]{geometry}
```

onde os parâmetros `top` e `bottom` podem ser vistos no desenho na Fig. **1.4**.

A orientação do documento em `portrait` ou `landscape` também pode ser facilmente alterada ao declarar nas opções do pacote.

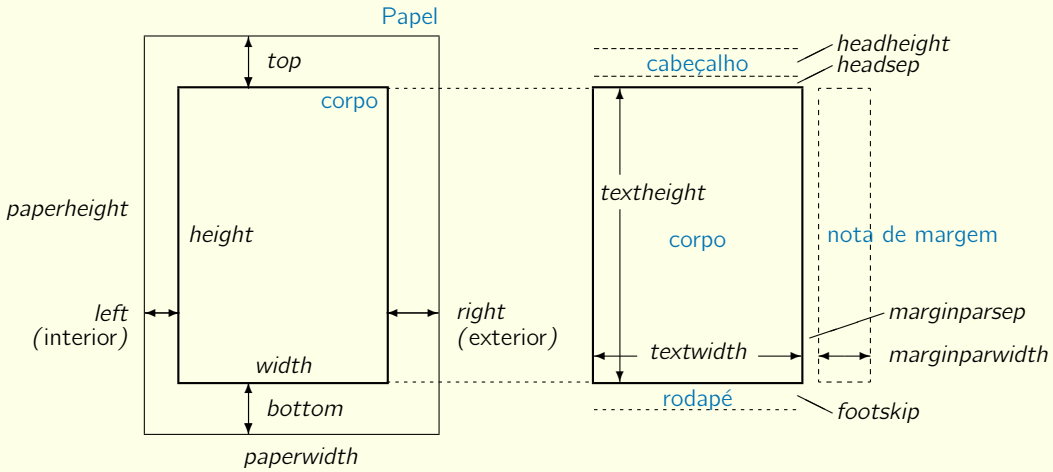


Figura 1.4: Parâmetros para geometry. Adaptado do manual.

1.7 setspace

Com este pacote modificamos o espaçamento entre as linhas dos parágrafos contribuindo para a organização e legibilidade do documento. Este documento utiliza:

```
\usepackage{setspace}
\onehalfspacing.
```

1.8 cmbright

A fonte é outra modificação que contribui para a melhora da leitura. Este pacote modifica a fonte para **Computer Modern Bright**. Além de não possuir serifa ela é fina, minimalista. Junto com o espaçamento de parágrafo e a cor de fundo é mais um recurso para amenizar a experiência de leitura.

```
\usepackage{cmbright}
```

1.9 hyperref e cleveref

Estes são os pacotes que aprimoram a capacidade básica do LaTeX em lidar com referências de um documento. Sem acrescentar pacotes, o LaTeX apenas numera automaticamente referências. Quando usamos `hyperref`, elas se transformam em *hyperlinks*, similares aos *links* dos *web browsers*, dinamizando bastante a leitura, permitindo "saltos" dentro do documento; `cleveref` automaticamente reconhece o tipo de referência (se é uma figura, tabela, seção, etc.) e permite personalizar cada tipo.

O ponto ao qual se deseja referir, ou "ponto de destino" pode ser várias coisas: uma figura, tabela, seção, capítulo, equação, referência bibliográfica, uma página na Web, um arquivo... Pode também ser uma palavra qualquer no texto. Para referenciar figuras, equações, seções e capítulos, **ponha ao menos um par** de etiquetas com um texto que represente bem o que está sendo referenciado, como por exemplo:

`\label{tipo:etiqueta de partida}` $\Longrightarrow$ `\cref{tipo:etiqueta de destino}`

Na Página **5** há um exemplo de utilização de `\label` da Fig. **1.3**. Veja o Código **1.2**.

Código 1.2 - Implementação de figuras

```
1  \begin{figure}
2    \begin{center}
3      \includegraphics[scale=0.4]{images/pastas.png}
4      \caption{Pastas dedicadas a partes específicas do documento.}
5    \end{center}
6    \label{fig:pastas}
7  \end{figure}
```

Logo antes de encerrar o ambiente `figure`, na linha 6 declaramos o `\label{fig:pastas}`. Ao longo do texto, sempre que formos nos referenciar a esse *label*, declaramos `\cref{fig:pastas}`. Note que "fig" diz respeito ao tipo de objeto que se está referenciando, uma figura. Isso é apenas uma boa recomendação, não faz parte do código. Além disso, este mesmo *label* serve para referenciar a página que o contém, utilizando-se `\cpageref{}`:

Código 1.3

```
Na \cpageref{fig:pastas} há um exemplo de utilização de label...
```

Tratando-se de *hyperlinks*, usamos a declaração `\href` da seguinte maneira para sumir com *links*, especialmente os longos e desinteressantes:

Código 1.4

```
\href{www.google.com}{Google}
```

Resultado 1.3

Google

1.10 xcolor

Este pacote é a sua aquarela no LaTeX. Ele é usado para definir (e misturar, no percentual desejado) cores que podem ser descritas de diversas formas. A cor de fundo deste documento é uma modificação da cor "yellow" que foi definida como:

```
\definecolor{yellow}{RGB}{248,253,137}
```

e assim a palavra é um *alias*, ou seja, ela significa o código RGB ao lado para comandos relacionados a cor, como no pacote `pagecolor` apresentado em seguida.

Há um conjunto de *aliases* pré-configurados. Essa lista de cores **pode ser expandida segundo essa listagem** para 68 cores pré-configuradas utilizando a declaração

```
\usepackage[dvipsnames]{xcolor}.
```

1.11 pagecolor

Este pacote (e seu nome bastante sugestivo) define a cor das páginas. A cor mencionada anteriormente foi "diluída" com branco, na proporção de 40%. Declaramos

```
\usepackage{pagecolor}
\pagecolor{yellow!40!white}.
```

Colorir páginas pode ser vantajoso para trazer conforto aos olhos e até mesmo facilitar a leitura, **conforme este estudo**, por exemplo.

1.12 pdflscape

Este pacote modifica a orientação da página, caso o conteúdo não caiba adequadamente (tabelas ou figuras largas demais) numa página no formato *portrait*. Este pacote acrescenta ao pacote `lscape` a funcionalidade de girar a página de modo que o `.pdf` gerado exiba as páginas em *landscape* na orientação correta.

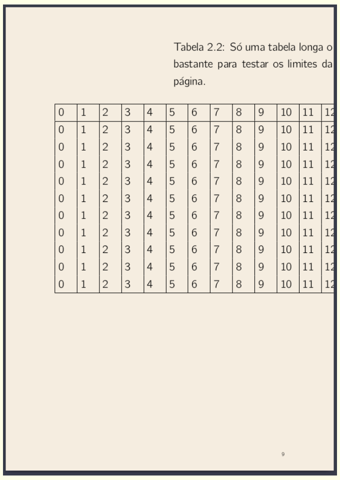


Tabela 2.2: Só uma tabela longa o bastante para testar os limites da página.

0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12

Figura 1.5: Sem usar o pacote...

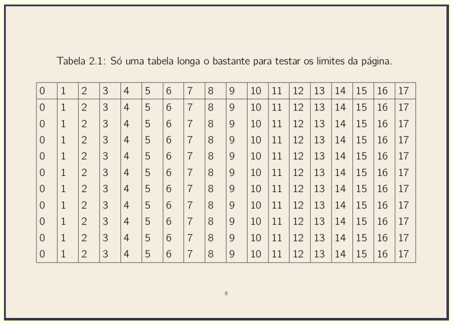


Tabela 2.1: Só uma tabela longa o bastante para testar os limites da página.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Figura 1.6: Usando o pacote.

1.13 biblatex

Este pacote faz o trabalho pesado de cuidar da bibliografia. É um pacote que permite muitas personalizações (e seu manual é bastante extenso), tornando ele a base para outros pacotes como o **biblatex-abnt**. Esse pacote carrega informações das referências a partir de um arquivo `.bib`. Ao longo da sua carreira de pesquisas é natural agregar várias referências, então é uma boa ideia utilizar **um programa** que as gerencie de modo mais fácil que alterar que o próprio arquivo `.bib`.

Cada referência no seu arquivo `.bib` tem um parâmetro que é o seu nome e o tipo de documento a que se deseja referenciar. *@booklamport1994latex* é o livro do Leslie Lamport de 1994 sobre LaTeX, por exemplo. Uma forma de utilizar referências é mostrada no Código 2.4.

As opções de customização do Biblatex são utilizadas pelas instituições para criar os seus próprios *estilos* de referências bibliográficas. Antes de fazer suas modificações, verifique os estilos pré-existentes. A página **Biblatex bibliography styles** contém demonstrações para vários estilos. Caso queira utilizar o estilo ABNT, inclua o parâmetro `style=abnt` na declaração do pacote.

Código 1.5

⇒ Insira os `\usepackage{}` no preâmbulo:

```
\usepackage[style=numeric,sorting=none]{biblatex}
  \addbibresource[]{./biblio/referencias.bib}
```

⇒ Este parâmetro no `hyperref` torna as citações vermelhas:

```
\usepackage[citecolor=red]{hyperref}
```

⇒ A declaração `cite` faz uso da referência no arquivo `.bib`:

```
A primeira referência é \cite{lamport1994latex}; a segunda referência é
\cite{kopka1995guide}.
```

⇒ Por fim o comando que gera a seção de referências:

```
\printbibliography
```

Resultado 1.4

A primeira referência é [1]; a segunda referência é [2].

[1] Leslie Lamport. *LATEX: a document preparation system: user's guide and reference manual*. Addison-wesley, 1994.

[2] Helmut Kopka e Patrick W Daly. "A guide to LATEX" . Em: *IEEE Multimedia* 2 (1995), pp. 473–480.

1.14 TikZ e PGF

Este pacote é um dos mais complexos, pois nos concede a capacidade de "desenhar" utilizando comandos. Dele derivam pacotes que permitem a criação de desenhos técnicos como o **CircuiTikZ** para desenhar circuitos ou **chemfig** para desenhar moléculas, até mesmo o **tikzpeople** para desenhar bonequinhos (mais inúmeros pacotes podem ser encontrados **aqui**). Entre esses inúmeros pacotes vale destacar o **pgfplots**, utilizado para gerar gráficos.

Desenhar usando TikZ é intuitivo, porém requer que sejamos um tanto prolixos no código, isso será visto no capítulo seguinte. A Fig. **1.7** é um exemplo de imagem desenhada no ambiente `tikzpicture`.

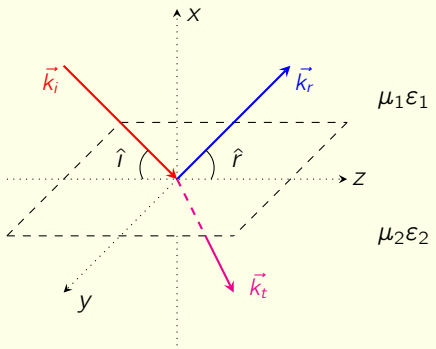


Figura 1.7: Reflexão de onda eletromagnética.

2. Definindo ambientes

Ao longo do seu código você irá declarar vários **ambientes** onde diversas coisas serão são inseridas no seu documento. Estes podem ser figuras, tabelas, listas, equações, referências bibliográficas, ajustes na formatação (como centralizar, ou modificar a orientação da página) e até inserção de outros `.pdf`. Na Seção **1.3** mostramos o ambiente `enumerate`, para gerar listas. Ambientes são declarados explicitamente pelo par `\begin{}` e `\end{}`.

Muitos ambientes são *floats*, ou seja, espaços que **não podem ser quebrados entre duas páginas**. Durante o uso dos ambientes declarações são feitas para configurá-los ao gosto do editor. Algumas delas servem para múltiplos ambientes, conforme será visto. LaTeX utiliza inglês para descrever

O ambiente para inserir figuras é um deles, vejamos como configurá-lo.

`figure`

Suponha que você deseja inserir figuras no seu documento. Isso se faz por meio da declaração

```
\includegraphics[]{}
```

Porém utilizar essa declaração fora do ambiente correto traz algumas restrições. Figuras normalmente são acompanhadas de legendas, e são referenciadas ao longo do documento. Então além da declaração acima é necessário definir um ambiente `figure` quando se deseja incluir imagens.

Código 2.1

```
\begin{figure}[!h]
  \includegraphics[scale=0.3]
  {images/TeX_lion.png}
  \caption*%
  {O leão chamado ``\TeX{}''?}
  \label{fig:leaoTeX}
\end{figure}
```

Resultado 2.4

O leão chamado "TeX"?

Inserir uma figura é o caso mais simples e às vezes até mesmo `\includegraphics[]{}` cumpre essa tarefa. Porém frequentemente necessitamos incluir diversas figuras em um pequeno aglomerado. Existem algumas maneiras para realizar isso. Vamos ver a mais simples.

`minipage`

Este ambiente é o mais básico quando se deseja posicionar objetos aglomerados. Com ele criamos um espaço dentro da página para acomodar uma mistura de "objetos" que podem ser texto, imagens, tabelas, listas, etc.

Código 2.2

```
\begin{figure}[h!]
  \begin{minipage}{0.5\textwidth}
    \centering
    \scalebox{-1}[1]{\includegraphics[width=0.2\paperwidth]
    {images/TeX_lion.png}}
    \caption{\reflectbox{\TeX{}}.}
    \label{fig:xet}
  \end{minipage}%
  \begin{minipage}{0.5\textwidth}
    \centering
    \includegraphics[width=0.2\paperwidth{images/TeX_lion.png}
    \caption{\TeX{}}
    \label{fig:tex}
  \end{minipage}
  \begin{minipage}{1\textwidth}
    \centering
    \captionsetup{type=figure}
    \includegraphics[width=0.2\paperwidth]{images/white_lion.jpeg}
    \captionof{figure}{Leão \emph{Moya}, na África do Sul.}
    \label{fig:moya}
\end{figure}
```

Resultado 2.2

Figura 2.1: XeTeX.

Figura 2.2: TeX

Figura 2.3: Leão *Moya*, na África do Sul.

Esse ambiente resolve o problema em diversos casos. Mas como ele é abrangente demais, não cobre especificidades desejadas ao lidarmos com múltiplas figuras. **Este documento** explica a evolução dos pacotes e aponta `subcaption` como sendo o pacote mais recente e mais completo.

`subcaption` Esse ambiente gerencia mais minuciosamente a inserção de imagens e tabelas também. Com ele é possível criar o que podemos chamar de "subfiguras". Diferente das Figs. **2.1** e **2.2**, que são independentes, subfiguras são elementos de uma figura maior. Ele interage perfeitamente com `hyperref`, permitindo referenciar-se precisamente a elas em **listas de figuras**.

Código 2.3

```
\begin{figure}[h!]
  \begin{subtable}[h]{0.5\textwidth}
    \centering
    \rowcolors{1}{yellow!75!black}{}
    \begin{tabular}{c|cc}
      Versão Ubuntu & Adjetivo & Animal\\
      \hline
      16.04 & Xenial & Xerus\\
      16.10 & Yakkety & Yakk\\
      17.04 & Zesty & Zapus\\
      17.10 & Artful & Aardvark\\
      18.04 & Bionic & Beaver\\
      18.10 & Cosmic & Cuttlefish\\
      19.04 & Disco & Dingo\\
      19.10 & Eoan & Ermine\\
      20.04 & Focal & Fossa
      \end{tabular}
    \caption{Versões do Ubuntu.}
    \label{tab:ubuntu}
  \end{subtable}
  \begin{subfigure}{0.5\linewidth}
    \centering
    \includegraphics[width=0.2\paperwidth]
    {unbionic_beaver.jpg}
    \caption{Castor não tão biônico.}
    \label{fig:unbionic}
  \end{subfigure}
  \caption{Ambiente \texttt{subfigure}.}
  \label{mix:subenv}
\end{figure}
```

E o resultado desse código:

Versão Ubuntu	Adjetivo	Animal
16.04	Xenial	Xerus
16.10	Yakkety	Yakk
17.04	Zesty	Zapus
17.10	Artful	Aardvark
18.04	Bionic	Beaver
18.10	Cosmic	Cuttlefish
19.04	Disco	Dingo
19.10	Eoan	Ermine
20.04	Focal	Fossa

(a) Versões do Ubuntu.

(b) Castor não tão biônico.

Figura 2.4: Ambiente `subfigure`.

Note que podemos referir a Figs. **2.4**, **2.4a** e **2.4b** separadamente. Uma vez que o ambiente maior definido na linha 1 do Código **2.3** é um ambiente de Figura e a tabela está dentro dele, ela é considerada uma figura pelo pacote `cleveref`.

`table` Este é o ambiente específico para gerar tabelas como Fig. **2.4a**. Funciona de modo semelhante ao ambiente `figure` quando queremos inserir uma `tikzpicture`. Porém, o objetivo aqui é gerar uma tabela via "subambiente" `\begin{tabular}`, de modo semelhante a uma `\begin{tikzpicture}`. Tabelas em LaTeX podem ser confusas de ler e difíceis de escrever,

mas felizmente há sites para ajudar. Um bom site é o Tables Generator, seu uso é bem direto. Não deixe de explorar as configurações de cor e alinhamento.

> **!** As tabelas não devem ter separadores verticais.

Para auxiliar a leitura, pode-se fazer alternância de cores das linhas. Há um exemplo na linha 4 do Código 1.3:

```
\rowcolors{1}{yellow!75!black}{}
```

Essa declaração informa que as cores das linhas devem se alternar entre *yellow!75!black* e *a cor de fundo* do documento, representada pelas {} vazias. O número "1" indica que a alternância começa da primeira linha da tabela. Reveja a Fig. **2.4a**.

> **!** Por vezes a tabela não se encaixa numa página.

O recurso é criar um escopo do tipo `scalebox` englobando o "subambiente" `tabular`. Veja como seria aplicado na Fig. **2.4a**

```
1 \scalebox{0.7}{
2   \begin{tabular}{c|cc}
3   Versão Ubuntu & Adjetivo & Animal\\
4   \hline
5   16.04 & Xenial & Xerus\\
6   ...
7   \end{tabular}
8   }
```

Aqui, `scalebox` irá ser aplicado ao ambiente `tabular`, reduzindo a tabela para 70% do seu tamanho.

`equation`

A capacidade de lidar com símbolos matemáticos foi um diferencial para o LaTeX ao longo dos anos. Notações compactas como a matemática necessitam de muitos símbolos em diversas posições, e os ambientes de matemática existem para isso.

A forma mais simples de usar símbolos matemáticos é em uma linha ao longo do texto. Para isto, devemos englobar o texto por um par de "$":

Código 2.4

```
1   $ hip^2 = cat_1^2 + cat_2^2 $
```

Resultado 2.4

$$hip^2 = cat_1^2 + cat_2^2$$

Porém qualquer expressão maior que uma linha (ou contendo frações, ou símbolos grandes como somatórios, integrais, etc) merece o seu próprio ambiente:

Código 2.5

```
1 \begin{equation}
2   \label{eq:continuedfrac}
3   \cfrac{1}{\sqrt{2}+
4     \cfrac{1}{\sqrt{2}+
5       \cfrac{1}{\sqrt{2}+\dotsb
6   }}}
7 \end{equation}
```

Resultado 2.5

$$x = \cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}} \tag{2.1}$$

O conteúdo desse **Short Math Guide** mostra muitas formas de declarar um ambiente de matemática, dependendo do posicionamento desejado para as suas equações. Tem também uma extensa listagem de símbolos para consulta.

`tikzpicture` Este é o ambiente genérico do pacote `tikz` para desenhar figuras genéricas. Quando colocado dentro de um ambiente `figure`, funciona de maneira muito parecida como quando incluímos uma figura usando `includegraphics`.

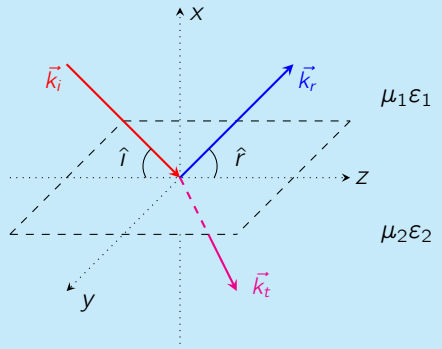
Pacotes derivados do `tikz` podem utilizar este ambiente ou definir um ambiente próprio, como é o caso do **CircuiTikZ** e seu `\begin{circuitikz}`.

O Código **2.6** e o Resultado **2.6** ilustram a utilização desse ambiente. Os espaços no código abaixo foram deixados para facilitar a leitura, mas o compilador os ignora, eles não são necessários.

Código 2.6

```
\begin{figure}[h!]
  \centering
  \begin{tikzpicture}[>= stealth, scale=1.5]
    % Plano XOY:
    \draw [dashed] (0,0) -- (2,0);
    \draw [dashed] (0,0) -- (1,1);
    \draw [dashed] (1,1) -- (3,1);
    \draw [dashed] (3,1) -- (2,0);
    % Eixos e seus nomes:
    \draw [dotted,->]  (1.5,0.5) -- (1.5,2)
    node [very near end, above=2mm, right] {$x$}; % Nome do eixo
    \draw [dotted] (1.5,0) -- (1.5,-1);           % O eixo
    \draw [dotted,->] (1.5,0.5) -- (0.5,-0.5)     % Seta do eixo
    node [very near end, right=1mm, below=1mm] {$y$};
    \draw [dotted,->] (0,0.5) -- (3,0.5)
    node [very near end, right=5mm] {$z$};
    \draw [thick,red, ->] (0.5,1.5) -- (1.5,0.5)
    node [very near start, above, left=1mm] {\small{$\vec{k_i}$}};
    \draw [thick,blue,->] (1.5,0.5) -- (2.5,1.5)
    node [very near end, above, right=1mm] {\small{$\vec{k_r}$}};
    \draw [thick,magenta,dashed] (1.5,0.5) -- (1.75,0);
    \draw [thick,magenta,->] (1.75,0) -- (2,-.5)
    node [very near end, below, right=1mm] {\small{$\vec{k_t}$}};
    % Arcos dos ângulos entre os raios refletidos e o plano:
    \draw (1.8,0.5) arc (-30:50:0.2)              % Arco
    node [below=1mm, right=2mm] {$\hat{r}$};      % Nome do ângulo
    \draw (1.2,0.5) arc (210:130:0.2)
    node [below=1mm, left=2mm] {$\hat{\imath}$};
    \draw (3.5,1.2) node {$\mu_1\varepsilon_1$};
    \draw (3.5,0) node {$\mu_2\varepsilon_2$};
  \end{tikzpicture}
  \caption{Reflexão de onda eletromagnética.}
  \label{fig:reflexample}
\end{figure}
```

Resultado 2.6



Reflexão de onda eletromagnética.

Existem algoritmos que servem de "ponte" entre outras linguagens ou outros programas e LaTeX. Isso nos dá a vantagem de editar em LaTeX uma figura que seria muito mais complexa que a do Código **2.6**. Podemos citar:

Na linguagem R existe o pacote TikZDevice. Em R existem comandos para gerar gráficos, e esse pacote gera arquivos `.tex` destes gráficos.

Geogebra é outra plataforma que também permite exportar gráficos para LaTeX.

MATLAB e Octave são poderosos programas que facilmente produzem gráficos. Para convertê-los para o formato .tex utilize o *script* MATLAB2TikZ.