UNIVERSIDADE ESTUDUAL DO SUDOESTE DA BAHIA
DCE

Iniciação à Linguagem de Programação LaTeX

B. M. Acióly

Vitória Da Conquista, Março de 2009.

Sumário

Sumário **2**

1 Iniciando o LaTeX **1**
1.1 Preparando um arquivo de entrada 1
1.1.1 A entrada 1
1.1.2 Sentenças e parágrafos 1
1.1.3 Traços 2
1.2 O Documento 7
1.2.1 O document class 7
1.2.2 A “Página"do Título 8
1.2.3 Criando Seções 2
1.2.4 Um Comando para Seccionar 2
1.2.5 Apresentando o Material 3
1.2.6 Declarações 8
1.2.7 Executando o LaTeX 9

2 Aprofundando o LaTeX **11**
2.1 Alterando o Estilo do Tipo 12
2.2 Símbolos de Outras Linguagens 14
2.3 Fórmulas Matemáticas 14
2.3.1 Algumas Estruturas Mais Comuns 14
2.3.2 Símbolos Matemáticos 16
2.4 Letras Gregas 16
2.5 Letras Caligráficas 16
2.6 Alguns Símbolos Matemáticos 17
2.7 Funções como log 18
2.8 Vetores 19
2.8.1 O Ambiente array 19
2.8.2 Delimitadores 20
2.9 Fórmulas Múltiplas 21
2.10 Pondo uma Coisa Acima da Outra 22
2.10.1 Sobre e Sob-linhas 22
2.10.2 Empilhando Símbolos 22
2.11 Espaço no Modo Matemático 23

2.12 Alterando o Estilo no Modo matemático 23
2.12.1 Estilos dos Tipos 23
2.13 Definindo Comandos e Ambientes 24
2.13.1 Definindo Comandos 24
2.14 Definindo Ambientes 25
2.15 Teoremas e Semelhantes 26
2.16 Figuras e Semelhantes 28
2.16.1 Figuras e Tabelas 28
2.17 Notas Marginais 28
2.18 Alinhando em Colunas 28
2.18.1 O ambiente Tabbing 29
2.19 O Ambiente Tabular 29
2.20 Simulando Texto Digitado 30

3 Circulando Informação 31
3.1 O Conteúdo 31
3.2 Referências Cruzadas 32
3.2.1 Primeiros resultados 32
3.3 Bibliografia e Citações 33
3.4 Usando a Biblioteca 34
3.5 Faça Por Si Próprio 35
3.6 Referências 36
3.7 Separando as Entradas 37
3.8 Fazendo um Índice ou um Glossário 38
3.9 Compilando as Entradas 39
3.10 Produzindo um Índice ou Glossário por si Próprio 39

4 Outros tipos de Documentos 42
4.1 Livros 42
4.2 Transparências 43
4.3 Transparências e Capas 43
4.4 Observações 44
4.4.1 Imprimir Somente Algumas Transparências e Notas 45
4.5 Cartas 45

Referências bibliográficas 48

Capítulo 1

Iniciando o LaTeX

1.1 Preparando um arquivo de entrada

A entrada em LaTeX é um arquivo de texto. Para criar um arquivo novo use um ambiente que suporte o LaTeX

1.1.1 A entrada

Muitos comandos em LaTeX descreve a estrutura lógica do documento. Para entender esses comandos, devemos saber como o LaTeX percebe essa estrutura lógica. Um documento contém estruturas lógicas de tamanhos diferentes, desde capítulos até letras individuais. Começaremos considerando uma estrutura intermediária mais familiar: sentenças e paráragrafos.

1.1.2 Sentenças e parágrafos

Descrever sentenças simples e parágrafos para o LaTeX não trás nenhum problema: digitamos naturalmente:

> O fim das palavras e sentenças são marcadas por espaços. Não existe nenhum problema como os espaços serão digitados. Um espaço é suficiente, e tem o mesmo efeito que 100 espaços.

```
O fim das palavras e sentenças são marcadas por espaços. Não existe
nenhum problema como os espaços serão digitados. Um espaço é
suficiente, e tem o mesmo efeito que 100 espaços.

Uma ou mais linhas em branco denota o fim de um parágrafo.
```

Uma ou mais linhas em branco denotam o fim de um parágrafo.[1]

Portanto, TeX ignora a maneira como é formatada uma entrada, ele só presta atenção aos conceitos lógicos: fim-da-palavra, fim-da-sentença e fim-do-paragrafo.

Aspas

Os textos digitados só usam dois tipos de aspas: Uma aspa dupla, "e uma aspas simples '. Esta última serve, também, como apóstrofe, quando o texto for, por exemplo, escrito em Inglês. O texto impresso, entretanto, usa uma versão esquerda e direita de cada, usando quatro símbolos diferentes. TeX interpreta o caractere ‘ como uma aspa simples à esquerda e ’, como uma aspa simples à direita. Para obter a aspa dupla digitar a aspa simples duas vezes.

```
A ‘convenção’ prescreve que a pontuação venha sob aspas,
como ``isto," no entanto, eu prefiro ``isto".
```

A ‘convenção’ prescreve que a pontuação venha sob aspas, como “isto,"no entanto eu prefiro “isto".

1.1.3 Traços

Podemos produzir três tipos diferentes de traços, digitando o caractere “-":

```
Uma palavra com traço-de-união ou hífen, com em raio-X.
Um traço no meio como em 1-2.
Um traço de pontuação, -, como isto.
```

Uma palavra com traço-de-união ou hífen, como em raio-X.

Um traço no meio como em 1-2.

Um traço de pontuação, -, como isto.

Em geral, não existe espaço após um traço, a não ser no último caso. O sinal menos não é um traço. Ele deve aparecer somente em fórmulas matemáticas, que discutiremos mais adiante.

O espaço após um período

Às vezes, desejamos colocar um pequeno espaço extra após um período ou uma sentença. Para o TeX não é fácil decidir o que se passa na cabeça do digitador. Por isso, ele assume que algumas pontuações terminam uma sentença, não colocando nenhum espaço extra.

[1] Sempre que apresentarmos um fragmento de texto em destaque, que, em geral, é uma explicação de como descrever para o LaTeX em seguida apresentaremos o mesmo fragmento, porém com formato e carateres diferentes, para indicar como será a apresentação desse fragmento quando impresso.

```
Aqui vai um exemplo. Não existe espaço extra.
```

Aqui vai um exemplo. Não existe espaço extra.

Se você desejar um espaço após um período, primeiro deixe uma linha em branco e em seguida use o comando **hspace{3ex}**, para um espaço de, por exemplo, três letras *x*.

```
Aqui vai um exemplo.

\hspace{3ex} Mas, é possível usar medidas como cm,
voltaremos a este assunto mais adiante.
```

Aqui vai um exemplo.

Mas, é possível usar medidas como cm, voltaremos a este assunto mais adiante.

Símbolos especiais

No TEX existem 12 caracteres especiais. Isto signfica que você só pode digitá-los usando comandos. Alguns deles, como $, representam símbolos que são muito usados num documento com objetos matemáticos. Sete desses símbolos podem ser produzidos usando a barra-inclinada-esquerda, \, em frente do caracteres correspondente.

```
\$ \& \% \# \_ \{ \} são fáceis de produzir.
```

$ & % # _ { } são fáceis de produzir.

Os outros três caracteres especiais (e \) usualmente só aparecem se forem simulados pelo teclado. Mais adiante voltaremos a esse assunto.

Comandos para gerar um texto simples

Uma parte de uma sentença pode ser produzida por um comando-de-gerar-texto. Por exemplo, os logos TEX e LATEX são produzidos pelos comandos *TeX* e *LaTeX*, respectivamente.

```
Algumas pessoas usam o \TeX\  plain, mas eu prefiro o \LaTeX.
```

Algumas pessoas usam o TEX plain, mas eu prefiro o LATEX.

Um comando útil de gerar texto, por exemplo, *ldots* produz três pontinhos, . . . , que serve para denotar algo que pretendemos omitir.[2]

```
Se estiver explícito \ldots , não é interessante.
```

[2]Simplesmente digitando três pontinhos não produz o espaço entre os pontos correto.

Se estiver explícito . . . , não é interessante.

Muitos dos comandos que vimos até aqui, consiste de um \(barra-inclinada-esquerda) seguido de um único nome, diferente de uma letra. A partir de agora, muitos comandos que encontrarmos tem nomes consistindo de uma \ seguida por uma ou mais letras. Ao ler o arquivo de entrada TeX sabemos que é o fim de um nome de comando quando ele encontrar uma não-letra: um dígito como “7", um caractere de pontuação como “;", um caractere especial como \, um espaço ou o fim de uma linha. Uma maneira mais comum de terminar esta espécie de nome de comando é com um espaço ou fim da linha. Assim, TeX ignora todos espaço seguindo ele. Se você quizer um espaço após o logo produzido pelo comando *LaTeX*, você não deve, simplesmente, digitar um espaço após o nome do comando, todo espaço como esse é ignorado. Você deve dizer a TeX para colocar o espaço, digitando o comando \ seguido de um espaço.

```
Esta página de \TeX\ foi produzida em \today\.
```

Esta página de TeX foi produzida em 11 de abril de 2009.

Observe que, agora, TeX ignora os espaço após o comando \today, na entrada e não produz qualquer espaço após os dados na saída.

O caso das letras num nome de comando é importante. Digitando \Today produz um erro, pois o nome do comando correto é *today*. Muitos nomes de comandos só tem letras minúsculas.

Enfatizando um texto

Os textos destacados são, normalmente, sublinhados num documento digitado e *italizado* no documento impresso. Sublinhar e italizar são conceitos visuais. Quando digitamos um documento, devemos nos preocupar somente com os conceitos lógicos de ênfase. O comando *emph* instrui TeX de que o texto é para ser enfatizado.

```
Aqui vai um \emph{texto enfatizado} sem graça.
```

Aqui vai um *texto enfatizado* sem graça.

No comando *emph{texto enfatizado}*, *emph* é um nome de comando e *texto enfatizado* é seu *argumento*. Muitos comandos não possuem argumentos, como *ldosts*, ou possuem um único argumento, como *emph*. Entretanto, existem alguns com argumentos múltiplos, todos esses argumentos dentro de parênteses. Os espaços são ignorados entre os argumentos e entre o nome do comando e seus argumentos.

Os comandos como *emph* podem ser aninhados um dentro do outro de modo óbvio. Muitos estilos usam o tipo romano ordinário para enfatizar o texto que aparece dentro do texto enfatizado.

```
Podemos ter \emph{ texto enfatizado \emph{dentro} texto enfatizado},
também.
```

Podemos ter, também, *um texto enfatizado* dentro *de um texto enfatizado*.

Evitando quebras de linhas

Ao colocar o texto no papel, um parágrafo pode ser quebrado em linhas de impressão. O texto se torna difícil de ler, se uma única unidade lógica for quebrada em linhas de forma arbitrária. Às vezes, uma linha não deve ser quebrada entre ou dentro de palavras. O digitador humano reconhece essas situações, mas TEX deve ser instruído sobre isto.

A quebra de linha deve ser evitado em certos espaços interpalavras. Por exemplo, a expressão “Capítulo 3"parece extranho, se “Capítulo"termina numa linha e “3"noutra. Digitar ~(o caratere til) produz um espaço de interpalavra no TEX que nunca quebra em linha. Abaixo, estão alguns exemplos indicando quando deve ser usado um ~.

```
Sr. ~João      Figura ~7   (1)~gnates
U.~S.~Grant    de 1 a ~10
```

É melhor não quebrar uma linha dentro de certas palavras. Por exemplo, devemos tentar evitar separar um nome(especialmente nome próprio). O comando instrui o TEX para imprimir o argumento inteiro na mesma linha. Por exemplo,

```
Doutor \mbox{Lamport}, não é?
```

Doutor Lamport, não é?

Notas de Rodapé

As notas de rodapé(footnote) são produzidos com um comando $\backslash footnote$, tendo o texto do rodapé como argumento.

```
Gnus\footnote{Um gnu é um grande animal} pode muito bém ser um gnusence.
```

Gnus pode muito bém ser um gnusense. .

.

.

Um Gnu é um grande animal.

Não existe espaço entre o Gnu e o $\backslash footnote$ neste exemplo. Adicionar um espaço colocaria o espaço não desejado entre o texto e a marca da nota de rodapé(o^1).

Um comando \ não pode ser usado como argumento em muitos comandos. Mais adiante, explicaremos como as notas de rodapé aparecem nos comandos com argumentos.

Fórmulas

Quando escrevemos um documento técnico, provavelmente ele contém fórmulas matemáticas. Uma fórmula que aparece no meio de uma sentença deve estar dentro dos comandos \(e \).

```
A fórmula \( x - 3y = 7 \) é fácil de digitar.
```

A fórmula $x - 3y = 7$ é fácil de digitar.

Qualquer espaço que digitarmos numa fórmula é ignorado.

```
\( x + y  \) é sempre igual a \(y+x\)?
```

$x + y$ é sempre igual a $y + x$?

TeX vê uma fórmula como uma palavra, que pode ser quebrada num certo ponto. O espaço antes de \(e após \) é tratado como uma separação inter-palavras comum.

Os subescritos são produzidos pelo comando _{ } e o superescrito pelo comando ^{ }.

```
\( a_{1} > x^{2n} / y^{2n} \)
```

$a_1 > x^{2n}/y^{2n}$

Esses dois comandos só podem ser usados dentro de uma fórmula matemática.

Quando usamos uma fórmula matemática, o caractere de aspa direito ' produz um traço($'$). Dois numa linha produz um traço duplo, e assim por diante.

```
... \( x' < x'' - y'_{3} < 10x''' z \)
```

... $x' < -x'' - y'_3 < 10x'''z$

As fórmulas matemáticas podem se tornar muito complexas. No próximo capítulo descreveremos muitos comandos adicionais para produzi-las. Aqui, consideraremos o uso de fórmulas no texto. Uma fórmula é uma unidade gramatical. Ela deve ser tratada como tal na estrutura da sentença.

```
A fórmula \( a<7 \) é um nome nesta sentença. Às vezes, ela é usada ...
```

A fórmula $a < 7$ é um nome nesta sentença. Às vezes, ela é usada ...

Começar uma sentença com uma fórmula torna-se difícil encontrar o início da sentença. Não faça isso. O melhor é usar uma fórmula como um nome. Ele, certamente, nunca aparece como uma sentença completa no texto que está sendo executado.

Uma variável como x é uma fórmula. Para digitar isso e obter a apresentação como tal, LaTeX trata $ ...$ da mesma maneira que \(... \).

```
Seja $x$ um primo tal que $ y > 2x $.
```

Seja x um primo tal que $y > 2x$.

Aconselhamos o uso de $ ... $ somente para fórmulas curtas, com uma única variável. É fácil esquecer dos caracteres $ que contorna uma fórmula longa. Podemos também digitar

```
\begin{math}  ... \end{math}
```

no lugar de \(... \). Poderemos querer usar esta forma para as fórmulas muito longas.

Ignorando a Entrada

Quando TEX encontra um caractere % na entrada ele ignora % e todos os caracteres seguindo-o na linha, incluindo o caractere de espaço no fim da linha. TEX também ignora espaço no início da próxima linha.

```
Gnus e armadi%dilhas são generalizadas ...
```

Gnus e armadi

O % tem dois usos: encerrar uma linha sem produzir qualquer espaço na saída[3] e colocar um comentário(uma observação para você mesmo) no arquivo de entrada.

1.2 O Documento

Agora, vamos sair das unidades lógicas intermediárias para aquela unidade maior: todo o documento. O texto de cada documento começa com o comando *\begin{document}* e termina com um *\end{document}*. A parte do arquivo de entrada precedendo o comando *\begin{document}* é chamado o *preâmbulo*.

1.2.1 O document class

O preâmbulo começa com um comando *\documentclass* cujo argumento é uma das classes predefinidas do documento que LATEX conhece. O arquivo *sample2e.tex* começa com

```
\documentclass{article}
```

que seleciona a *classe artigo*. A outra classe LATEX padrão, usada para os documentos comuns é a classe *report*. A classe *article* é usada, geralmente, para documentos mais curtos que a classe *report*. As outras classes padrão são descritas numa lição mais adiante.

Além de escolher a classe, você pode querer selecionar algumas opções, dentre as classes de documentos. As opções para as classes *article* e *report* incluem os seguintes:

11pt Especifica um tamanho de tipo conhecido como *onze pontos*, que é dez por cento maior que o *dez pontos*, o tipo normalmente usado.

12pt Especifica um tamanho do tipo *doze pontos*, que é vinte por cento maior que os dez pontos.

twoside Formato da saída para impressão em ambos os lados da página. (LATEX não tem cotrole sobre a impressão).

twocolumn Produz saída em duas colunas.

[3] No entanto, não podemos separar um nome de comando em duas linhas.

As outras opções serão descritas mais adiante.

Você especifica uma opção *document class* colocando entre chaves imediatamente após "*documentclass*", como em

```
\documentclass[twoside]{report}
```

As múltiplas opções são separadas por vírgulas.

```
documentclass [twocolumn,12pt]{aticle}
```

Não deixe nenhum espaço dentro dos colchetes.

O comando *documentclass* pode ser usado com ou sem a parte de opção. As opções, colocadas dentro de colchetes, são um argumento opcional do comando. É uma convenção em LaTeX que os argumentos opcionais estejam dentro de colchetes e os argumentos principais estejam dentros de chaves. O TeX ignora um espaço após um nome de comando como *documentclass* e entre os argumentos do comando.

A classe *document class* define os comandos para especificar a estrutura lógica do LaTeX padrão. As estruturas adicionais são definidas por *pacotes(packages)*, que são carregados pelo comando *usepackage*. Por exemplo,

```
\usepackage{latexsym}
```

carrega o pacote *latexsym*, que define comandos para produzir certos símbolos matemáticos especiais. Um pacote pode ter opções especificadas por um argumento opcional de *usepackage*, exatamente como uma para *documentclass*.

Você provalmente quer definir alguns comandos novos para uma estrutura especial, usada em seu documento particular. Por exemplo, se você quizer escrever sobre culinária, provavelmente vai querer definir seus próprios comandos para formatar receitas, como veremos mais adiante. Essas definições devem estar no *preâmbulo*, após os comandos *documentclass* e *usepackage*. O *preâmbulo* pode, também, conter comandos para alterar alguns aspectos do formato. Se você tiver comandos ou alterações de formatos que usa em vários documentos, você pode querer definir seu próprio pacote, como descreveremos em outro capítulo.

1.2.2 A "Página"do Título

Normalmente, um documento tem uma "página" do título apresentando seu título, um ou mais autores e uma data. Escrevemos página sob aspas porque, para um pequeno documento, essa informação pode ser listada na primeira página do texto, em vez de numa página especial. A informação do título consiste do próprio título, do autor(es) e da data. Ele é especificado por três declarações: *title*, *author* e *date*. A "página"do título atual é gerado por um comando *maketitle*.

```
\title{Gnus do mundo}
```

```
\author{R. Dather \and J. Pennings \andB. Talkmore}

\date{10 de outubro de 1999}

...

\maketitle

    ...
```

Gnus do mundo

R. Dather J. Pennings B. Talkmore

10 de ouubro de 1999

Observe como os autores múltiplos são separados pelos comandos *\and*

O comando *\maketitle* vém após o *\begin{document}*, *usualmente, antes de qualquer outro texto. Os comandos \title, \author e \date podem vir em qualquer lugar antes de \maketitle. O \date é opcional. LaTeX supre a data corrente se a declaração for omitida, mas o \title e o \author devem aparecer, se um comando \maketitle for usado. Os comandos para acrescentar outras informações, como o endereço do autor e os agradecimentos serão descritos noutro capítulo.*

1.2.3 Criando Seções

As sentenças estão organizadas em parágrafos e os parágrafos, por sua vez, estão organizados numa estrutura de seções *hierárquicas. No momento, você está lendo a seção 2.3, entitulada criando seções, da primeira lição. Usaremos o termo* unidades secionais *para raciocinar sobre capítulos, seções e subseções.*

Uma unidade secional começa com o título da unidade como seu argumento.

```
\subsection{Uma comando para seccionar}
\LaTeX\ automaticamente gera o número da seção. As linhas em branco
antes ou após um comando para seccionar não tem efeito.
```

1.2.4 Um Comando para Seccionar

LaTeX automaticamente gera o número da seção. As linhas em branco antes ou após um comando para seccionar não tem efeito.

A classe do documento determina quais comandos de seccionar são providenciados, a classe padrão tem as seguintes.[1]

```
\part        \subsection        \paragraph
\chapter     \subsubsection     \subparagrph
\section
```

A classe de documento article não contém o comando \chapter(capítulo), que torna fácil de um "artigo", com um capítulo de um "relatório"(report). O exemplo, acima, como muitos outros, que teremos oportunidade de ver, assume a classe de documento article, o 4.7 indicando que esta é a sétima subseção, da seção 4. Na classe report, esta subseção deve ser numerada 5.4.7, com o "5" sendo o número do capítulo.

A unidade seccional denotado por cada um desses comandos deve aparecer como uma subunidade de uma denotada pelo comando precedente, exceto que o uso de \part é opcional. Uma subseção deve ser parte de uma seção, que na classe report deve ser parte de um capítulo.

[1] Os nomes dos parágrafo e subparágrafo não são bons, pois eles denotam unidades que são frequentemente compostas de vários parágrafos. Esses nomes permanecem por razões históricas.

O comando \part é usado pelas principais divisões no documento. Ele não afeta a enumeração das unidades menores, na classe article, se a última seção, da Parte 1, for a seção 5. Então a primeira seção da parte 2 é a seção 6.

Se existir um apêndice, ele começa com o comando \appendix e usa o mesmo comando de secionar que a principal parte do documento. O comando \appendix não produz qualquer texto. Ele tem como efeito, simplesmente, que as unidades secionais sejam enumeradas apropriadamente por um apêndice.

A classe do documento determina a aparência do título, incluindo se ou não ele é enumerado. As declarações para controlar as seções numeradas são apresentadas mais adiante, em outro capítulo, onde também apresentaremos como fazer tabelas de conteúdos(índice).

O argumento de um comando de seção pode fazer mais que, simplesmente, produzir o título da seção. Discutiremos isso futuramente.

Quando consideramos a posição que um argumento aparece num arquivo de entrada, um comando em LaTeX pode ser frágil, significando que se o colocarmos em outro lugar, no arquivo de entrada, ele não funciona. Os comandos frágeis raramente são usados no argumento de um comando de seccionar. Dos comandos que introduzimos até aqui, os únicos que são frágeis são: \(, \), \begin, \end e \footnote. Provavelmente nenhum desses você venha a precisar no título de uma seção. Nas raras ocasiões que você colocar um comando frágil no título de uma seção, você, simplesmente, protege-o com um comando \protect. O comando \protect vém antes que todo nome de um comando frágil, como em:

```
\subsection{ \protect\( x + y \protect\)  é  primo? }
```

Isto, realmente, é um exemplo não muito bom porque $ não é comando frágil, podemos digitar também,

```
subsection{$x + y$ é primo?}
```

mas, porque é muito raro um problema como esse, é difícil encontrar um bom exemplo usando o comando descrito neste capítulo.

Um argumento no qual for necessário os comando frágeis precisam de \protect será chamado um argumento móvel. Os comandos que não forem frágeis serão chamados robustos. Para qualquer comando que, razoalvemente, podemos esperar usar um argumento móvil, indicaremos se ele é frágil ou robusto.

1.2.5 Apresentando o Material

Agora, retornamos ao nível das sentenças individuais . Uma sentença como

> Ele se virou e disse para mim, “Minha resposta é não!"e, então, se retirou.

contém uma sentença completa sob aspas. Um parágrafo completo, pode, mesmo, aparecer dentro de uma sentença, como em

> *Ele virou-se e me disse: "Fiz tudo que poderia ser feito". Me recuso a continuar fazendo parte dela". Minha resposta é não!", e então retirou-se.*

É difícil entender essa sentença, da maneira como ela está escrito. Entretanto, não existe problema se ela for lida em voz alta, usando um tom de voz diferente para as aspas. Uma analogia tipográfica para alterar o tom de voz é dispor o texto com indentação, chamado, também, formatação. *A sentença, acima, é muito mais fácil de ler quando tipografado como segue:*

> *Ele virou-se e me disse: Fiz tudo que poderia ser feito. Me recuso a continuar fazendo parte dela. Minha resposta é não, e, então, retirou-se.*

O material, agora, está distribuído logicamente como uma unidade de vários níveis, embora gramaticamente essa sentença possa fazer parte de outra sentença, como um todo, ou mesmo de vários parágrafos. Para decidir se uma porção do texto seria disposta numa unidade seccional separada, devemos determinar se ela está logicamente subordinada ao texto onde ela está contida ou funciona como uma unidade independente. Muitas vezes são usados distaques.

```
 ... exemplo de um pequeno distaque.
\begin{quote}
 \ldots\ é uma boa ideia colocar o arquivo de entrada
tão fácil de ler quanto possível.
\end{quote}
Ela está indentada em ambas as margens.
```

> *O seguinte é um exemplo de um pequeno distaque.*
>
> *... é uma boa ideia colocar o arquivo de entrada tão fácil de ler quanto possível. Ela está indentada em ambas as margens.*

Este exemplo ilustra um tipo de construção em LaTeX chamado um ambiente(environmt), *que é digitado*

```
\begin{nome} ... \end{nome}
```

onde nome é o nome do ambiente. O ambiente de aspas produz um disposição adequada para um curto destaque. Já tivemos oportunidade de encontrar dois outros exemplos de ambientes: O ambiente matemático *e o ambiente de* documento.

A classe de documento padrão LaTeX fornece ambientes para produzir várias espécies diferentes de formatação. As linha em branco antes ou após o ambiente marca um novo parágrafo. Portanto, uma linha em branco após o comando \end significa que o texto seguinte começa um novo parágrafo. As linhas em branco antes e após o ambiente signfica que ele é um parágrafo completo.

Destaques

LaTeX tem dois ambientes diferentes para fazer destaques. O ambiente quote é usado para um distaque curto ou uma sequência de destaques curtos separados por linhas em branco.

```
Nosso presidentes ... disseram
\begin{quote}
O desaforo vai parar aqui. \emph{Getúlio Vargas}

Não sou um cretino.   \emph{João Figueredo}

Não existe excesso ... \emph{Médice}
\end{quote}
```

Nosso pressidentes ... disseram

> *O desaforo var parar aqui.* Getúlio Vargas
> *Não sou um cretino.* João Figueredo
> *Não existe excesso ...* Médice

O ambiente quotation é usado para destaques de mais de um parágrafo. Como é comum, os parágrafos são separados por linhas em branco.

```
Aqui vai um aviso para lhe lembrar  que você
está usando \LaTeX:
\begin}{quotation}
  Os ambientes para fazer destaques
  podem ser usados para outras coisas.

  Muitos  problemas podem ser resolvidos via
  uma nova alicação de um  ambiente já existente.
\end{quotation}
```

> *Os ambientes para fazer destaques podem ser usados para outras coisas.*
> *Muitos problemas podem ser resolvidos via uma nova aplicação de um ambiente já existente.*

Listas

LaTeX fornece três tipos de construções de ambientes: itemize, enumerate e description. Em todos os três, cada item da nova lista começa com um comando \item. As listas itematizadas são feitas com o ambiente itemize e listas enumeradas com o ambiente enumerate.

```
\begin{itemize}
\item   Cada item da lista é marcado com um {\em rótulo}. Cada rótulo nesta lista
         itemizada são pontos.

\item Uma lista pode estar aninhada dentro de outra.
   \begin{enumerate}
   \item O rótulo do item numa lista enumerada são numerais ou letras.
   \item Uma lista deve ter, no mínimo, dois itens.
   \end{enumerate}
  O \LaTeX\ permite, no mínimo, quatro níveis de listas aninhadas, que são mais ques suficientes.
\item As linhas em branco antes de um item não tem efeito.
\end{itemize}
```

- *Cada item da lista é marcado com um* rótulo. *Cada rótulo nesta lista itemizada são pontos.*
- *Uma lista pode estar aninhada dentro de outra.*
  1. *O rótulo do item numa lista enumerada são numerais ou letras.*
  2. *Uma lista deve ter, no mínimo, dois itens.*

  O LaTeX permite, no mínimo, quatro níveis de listas aninhadas, que são mais que suficientes.
- *As linhas em branco antes de um item não tem efeito.*

Num ambiente de descrição, especificamos o rótulo do item com um argumento opcional para o comando \item, dentro de parênteses. (Embora o argumento seja opcional quando o item é omitido não fica iteressante.)

Três pontos importantes que prentendemos destacar:

```
\begin{description}
\item É preciso atentar para...
\item Tudo foi observado ...
\item Por último ...
\end{description}
```

É preciso atentar para...

Tudo foi observado ...

Por último ...

Se quisermos delimitar e/ou produzir a saída em parênteses devemos usar os carateres [e]. Isto pode causar alguma confusão, se o texto de um item começar com um [ou se um comando opcional de \item contém um colchete. Todos os comandos que tem um argumento opcional são frágeis.

Poesia

A poesia é construída num ambiente de verso. Uma nova estrofe começa com uma ou mais linhas em branco. As linhas dentro de uma estrofe são separadas pelo comando \\.

```
\begin{verse}
Existe um ambiente para o verso \\
cujas características alguns poetas seguem.

pois, em vez de fazê-los, \\
com  \emph{todas} as linhas  quebradas,\\
Ele permite ... ser conciso.
\end{verse}
```

Existe um ambiente para o verso
cujas características alguns poetas seguem.
pois, em vez de faze-los,
com todas *as linhas quebradas,*
Ele permite ... ser conciso.

*O comando * é o mesmo que \\, exceto que ele previne o LaTeX de começar uma nova página naquele ponto. Ele pode ser usado para prevenir um poema de ser quebrado em páginas de uma maneira confusa. O comando \\ e * são usados em todos os ambientes nos quais instruímos o LaTeX onde quebrar linhas. Vários desses ambientes serão descritos mais adiante. O comando * é chamado uma* forma-* *do comando \\. Vários outros comandos também tem sua versão forma-*.que é ligeiramente diferente da maneira ordinária. Essa versão é obtida colocando * após o nome do comando.*

Quase todos os comandos que da forma- são frágeis. O próprio comando na forma-* é frágil.*

Fórmulas Marcadas

Uma fórmula matemática é marcada quando ela é ou muito longa para se adequar confortavelmente num texto corrido ou tão importanto que queremos destacá-la para futuras referências. O LaTeX fornece os ambientes displaymath e equation para destacar fórmulas. Eles são iguais, exceto que as equações enumeram as fórmulas e displaymath não. Devido o fato de que as equações destacadas são tão frequentemente usadas em matemáticas, LaTeX permite digitar \[...\], em vez de

```
\begin{displaymath}  .. \end{displaymath}

Aqui vai um exemplo de uma equação destacada, mas não numerada:
\[ x'+ y^{2} = z_{i}^{2}  \]
e aqui vai a mesma equação numerada:
\begin{equation}
 x' + y^{2} = z_{i}^{2}
\end{end}
```

Aqui vai um exemplo de uma equações destacada, mas não numerada:

$$x' + y^2 = z_i^2$$

e aqui vai a mesma equação numerada:

$$x' + y^2 = z_i^2 \tag{1.1}$$

A classe documento determina como as equações são numeradas. Num capítulo posterior descreveremos como LaTeX pode, automaticamente, manipular referências a número de equações de modo que não tenhamos que nos precupar com números.

Uma fórmula destacada, como qualquer texto destacado, não deve começar um parágrafo. Ademais, ele não deve formar, por si próprio, um prágrafo completo. Essas duas observações são resumidas numa simples regras: na entrada nunca deixe uma linha em branco antes de uma fórmula destacada.

O TeX não quebrará a fórmula num ambiente displaymath ou equation numa linha. Noutro capítulo veremos como os comandos criam uma fórmula múlti linha ou uma sequência de fórmulas destacadas.

1.2.6 Declarações

Podemos enfatizar um fragmento grande de texto, como uma quotation. Podemos fazer isso com o comando \emph, mas isso torna o arquivo de entrada difícil de ler, pois temos de procurar a chave direita para ver onde o argumento termina. Além disso, é fácil, acidentalmente, remover a chave que fecha, quando editamos o texto. Podemos, em vez disso, usar o comando \em, que instrui TeX para começar o texto enfatizando.

```
Esta prova é muito insípida.
\begin{quote}
Espere! \em Aqui vai um destaque excitante.
\end{quote}
Não estamos alegres que tal excitamente tenha acabado?
```

Esta prova é muito insípida.

> *Espere!* Aqui vai um destaque excitante.

Não estamos alegres que tal excitação tenha acabado?

Como explicaremos, abaixo, o comando TeX \end{quote} tem como efeito suspender o texto que está sendo destacado.

Diferentemente de outros comandos que encontramos até aqui, \em não produz nem texto nem espaço. Em vez disso, ele afeta *a maneira como LaTeX imprime o texto que o segue. Um comando como esse é chamado uma* declaração. *Muitos aspectos da maneira como LaTeX formata um documento, o tipo estilo, como as margens são aumentadas e etc., são determinadas por declarações. A declaração \em instrui LaTeX para alterar o tipo estilo para um apropriado para indicar ênfase. O escopo de uma declaração termina por*

um comando \end ou uma chave a direita. Na entrada, as chaves e os comandos \begin e \end devem vir casados em pares. O escopo de uma declaração termina por um primeiro \end ou }, cujo par que casa, \begin ou {, precede a declaração. O seguinte exemplo mostra somente as chaves, os comandos \begin e \end(sem seus argumentos e as chaves, fechando esse argumento), e uma declaração \em de algum texto e entrada. O casamento de chaves e um casamento dos comandos \begin \end tem os mesmos números. A região começando no comando \begin$_4$ e terminando com }$_3$, é o escopo da declaração \em

$$\backslash begin_1 \quad \{_2 \quad \}_2 \ \{_3 \quad \backslash em \ \backslash begin_4 \ \{_5 \quad \}_5 \quad \backslash begin_4 \quad \}_3 \quad \backslash end$$

As chaves podem ser aquelas que delimitam um argumento de comando ou aquelas que são colocadas para delimitar o escopo de uma declaração, como na primeira ênfase de esse *em*[2]

```
Esse {\em esse} esse \emph{esse} ...
```

Esse esse *esse* esse ...

Os comandos \{ e \{ não contam para os propósitos de casamentos de chaves e delimitação dos escopos nas declarações.

Cada declaração tem um ambiente correspondente do mesmo nome(menos o caractere \). Digitando

```
\begin{em} ... \end{em}
```

é equivalente a digitar {\em...}. (Se uma declaração tem argumentos, eles se tornam argumentos adicionais do ambiente \begin correspondente.) Usar a forma de ambiente de uma declaração em vez do delimitador de seu escopo torna o arquivo de entrada mais fácil de ler.

1.2.7 Executando o LaTeX

Como executar um programa LaTeX depende do ambiente que você estiver usando. De qualquer modo quando usar seu próprio arquivo de entrada pela primeira vez, provalmente as coisas não acontecerão como o desejado. Certamente, haverão muitos erros. Mais adiante nos aprofundaremos na questão erro. No momento ajudaremos como abordar o erro e corrigir erros simples.

Consideremos a seguinte expressão $x + y = 4. Quando instruírmos o LaTeX para compilar, ele emite a seguinte mensagem:

```
Algumas observações ...

?
```

[2] Você deve usar o comando *\emph* em vez da declaração *\em* pequenos pedaços de textos pois ele produz um espaçamento melhor.

Para corrigir o erro que está anunciado em "Algumas observações ..."digite "e"e pressione enter. O compilador emTeX aponta a linha onde está o erro.

No caso, o erro é porque falta o $, a direita, para casar com o da esquerda.

Capítulo 2

Aprofundando o LaTeX

No capítulo anterior vimos a descrição de comandos para construir documentos simples. Neste capítulo pretendemos ser um pouco mais sofisticados em nossa formatação. Os comandos e ambientes descritos neste capítulo visam atender essa necessidade. Antes de começar vamos conhecer um pouco sobre como TeX opera.

Quando TeX processa um texto de entrada ele sempre o faz em um de três modos: mode de parágrafo, modo matemático ou o *modo esquerda-direita(*modo LR, para simplificar). *O modo de parágrafo é aquele normal em TeX quando ele processa os textos comuns. Neste modo TeX ver um texto de entrada como uma sequência de palavras cujas sentenças são quebradas em linhas, parágrafos e páginas.*

O TeX está no modo matemático quando ele está gerando uma fórmula matemática. Mais precisamente, ele entra no modo matemático quando ele encontra um comando como

```
$ \(, \[ ou $$,  \begin{equation},
```

que inicia uma fórmula matemática. Ele deixa o modo matemático quando os comandos correspondentes, como, por exemplo, \), *termina uma fórmula. Quando TeX está no modo matemático ele considera as letras no arquivo de entrada como símbolos matemáticos tratanto as, por exemplo, como o produto de a por s, e ignorando qualquer caractere de espaço entre eles.*

No modo LR, como no modo parágrafo, o TeX considera o arquivo de entrada como uma cadeia de palavras com espaços entre elas. Entretanto, diferentemente do modo parágrafo, ele nunca inicia uma nova linha. O comando *mbox tem como efeito fazer com que a cadeia de argumentos não sejam ao longo da linha.*

Modos diferentes podem ser aninhados um dentro do outro. Se colocarmos o comando *mbox dentro de uma fórmula matemática, TeX está no modo RL enquanto ele estiver processando os argumento e não no modo matemático. Por exemplo,*

```
\( y > z, \mbox{ se $x^{2}$ for real}. \)
```

$$y > z, \text{ se } x^2 \text{ for real.}$$

O TeX está no modo matemático quando estiver processando qualquer coisa fora de \mbox.

2.1 Alterando o Estilo do Tipo

O estilo do tipo é usado para indicar a estrutura lógica. Nestas notas, os textos enfatizados aparecem no tipo de estilo itálico.

Em LaTeX um estilo é especificado por três componentes: forma, série e família.

Os estilos mais comuns são:

1. *O tipo normal: Aquele que aparece no texto se nenhum estilo for solicidado.*

   ```
   \texup{Forma vertical  Geralmente a normal.}
   ```

   Forma vertical. Geralmente a normal.

2. *A forma itálica é usada normalmente para enfatizar o texto,*

   ```
   \textit{Forma itálica. Normalmente para enfatizar o texto.}
   ```

   Forma itálica. Normalmente usada para enfatizar o texto.

3. *Forma inclinada, que é um pouco diferente da forma itálica.*

   ```
   \textsl{Forma inclinada. Um pouquinho diferente da forma itálica,}
   ```

   Forma inclinada. Um pouquinho diferente da forma itálica,

4. *Forma maiúscula pequena.*

   ```
   \textsc{Forma maiúscula pequena,}
   ```

   FORMA MAIÚSCULA PEQUENA,

   ```
   \textmd{Série meio. Geralmente a normal,}
   ```

Série meio. Geralmente a normal,

```
\textbf{Série letras destacadas. Normalmente usadas em cabçários,}
```

Série letras destacadas. Normalmente usadas em cabeçários,

```
\textrm{Família romana,}
```

Família romana,

```
\texttt{Família digitador,}
```

`Família digitador,`

5. *Esses comandos podem ser combinados numa forma lógica para produzir uma ampla variedade de estilos de tipos.*

```
\textsf{\textbf{Quem \textmd{neste} mundo}} já \textit{\texttt {usou}}...
```

***Quem** neste **mundo** já* `usou`*...*

Alguns estilos tipográficos podem não estar disponíveis em nosso computador. Se especificarmos um estilo que não esteja disponível, o LaTeX escreverá uma mensagem de aviso e o substitui por um estilo similar.

Cada um dos comandos texto-estilo, descrito anteriormente, tem um comando de declaração correspondente. O destaque em negrito pode ser obtido com \textbf ou a declaração {\bf mais o texto}. O mesmo vale para os demais estilos.

```
\textbf{Mais} e {\bf mais} animais estão cruzando a estrada.
```

***Mais** e **mais** animais estão cruzando a estrada.*

Finalmente, observaremos que nenhum desses estilos podem ser usados no modo matemático. O estilo tipográfico é uma propriedade visual. Os comandos para especificar as propriedades visuais não pertencem ao texto, mas às definições dos comandos que descrevem as estruturas lógicas. LaTeX fornece o comando \emph para enfatizar um texto.

2.2 Símbolos de Outras Linguagens

O TeX foi projetado para textos em língua inglêsa. Outras línguas que não o inglês, possuem vários símbolos como acentos e outros. Para obter os acentos em português devemos incluir no preâmbulo de nosso programa o seguinte comando:

```
\usepakage[portuges]{babel}
```

que produz qualquer acento da língua portuguêsa. Para símbolos de outras línguas recomendamos um manual de LaTeX .

2.3 Fórmulas Matemáticas

Uma fórmula que aparece no texto de execução, chamada fórmula **in-texto**, *é produzido pelo ambiente* math. *Este ambiente pode ser invocado por uma das duas formas curtas: \(... \) ou $...$, assim como pela construção usual \begin ...\end. O ambiente* displaymath, *que tem a forma curta \[...\], produz uma fórmula matemática destacada, porém não numerada. Uma fórmula destacada e numerada é produzida pelo ambiente* equation, *nomeadamente,*

```
 \begin{equation}....\end{equation}.
```

```
\begin{equation}
 y = 2x
\end{equation}
```

$$y = 2x \tag{2.1}$$

Os ambientes math, displaymath *e* equation *colocam o TeX no modo matemático.*

2.3.1 Algumas Estruturas Mais Comuns

Sub-escrito(índice) e Super-escrito(expoente)

Os sub-escritos e super-escritos são feitos com os comandos _ e ^.

Esses comandos podem ser combinados para obter expressões com sub-escritos e super-escritos mais complexas.

```
x^{2y}  x^{y^{2}}  x^{y}_ {1}

x_ {2y}  x^{y_{1}} x_{1}^{y}
```

$$x^{2y} \quad x^{y^2} \quad x_1^y$$

$$x_{2y} \quad x^{y_1} \quad x_1^y$$

Frações

As frações que forem denotadas com o símbolo / são feitas usando o teclado como, por exemplo,

```
multiplicando por $1/2$ resulta \( (m+n)/n. \)
```

multiplicando por $1/2$ *resulta* $(m+n)/n$.

Muitas frações no texto a ser executado são escritas dessa maneira. Usamos o comando $\backslash frac$ *quando pretendemos que o traço de fração seja horizontal. Ele tem dois argumentos: o numerador e o denominador.*

```
\[ x = \frac{y + z/2}{y^{2} + 1} \]

\[ \frac{x+y}{1 + \frac{y}{z+1}} \]
```

$$x = \frac{y + z/2}{y^2 + 1}$$

$$\frac{x+y}{1+\frac{y}{z+1}}$$

Raízes

O comando $\backslash sqrt$ *produz a raíz quadrada de seus argumentos. Ele tem um argumento opcional para as outras raízes. Ele é um comando frágil.*

```
...\(\sqrt{x+y} \) ... \( \sqrt[n]{2} \)...
```

...$\sqrt{x+y}$ ... $\sqrt[n]{2}$...

Elípses

O comando $\backslash dots$ *e* $\backslash cdots$ *produzem duas espécies de elípses(...)*

```
Uma elípse baixa: $x_{i}, \ldots , x_{n}.$

Outra centrada: $a + \cdots + z.$
```

Uma elípse baixa: $x_i, \ldots, x_n$.

Outra centrada: $a + \cdots + z$.

Use $\backslash ldots$ *entre vírgulas e símbolos justapostos como* $a...z$. *Use* $\backslash cdots$ *entre símbolos como* $+, -$ *e* $=$. *TEX também pode produzir elípses vertical e horizontal, que são usados principalmente em vetores.*

```
$\vdots$
```

$\vdots$

```
$\ddots$
```

$\ddots$

2.3.2 Símbolos Matemáticos

Existem comandos em TeX para quase todos, senão todos, os símbolos matemáticos que precisemos usar. Relembramos que eles somente podem ser usados no modo matemático.

α \alpha	θ \theta	o o	τ \tau
β \beta	ϑ \vartheta	π \pi	υ \upsilon
γ \gamma	ι \iota	ϖ \varpi	ϕ \phi
δ \delta	κ \kappa	ρ \rho	φ \varphi
ε \epsilon	λ \lambda	ϱ \varrho	χ \chi
ε \varepsilon	μ \mu	σ \sigma	ψ \psi
ζ \zeta	ν \nu	ς \varsigma	ω \omega
η \eta	ξ \xi		

Tabela 2.1: Tabela das letras gregas

2.4 Letras Gregas

Os comandos para produzir uma letra grega minúscula é obtido acrescentando(antes) \ ao nome da letra. Para conseguir uma letra grega maiúscula basta colocar em maiúscula a primeira letra do nome do comando.

```
É fácil fazer letras gregas como $\pi$ e $\Pi$
```

É fácil fazer letras gregas como π *e* Π.

Na tabela 2.2 está o alfabeto grego minúsculo e maiúsculo. Observe que usamos o comando \var iniciando algumas letras para obter uma variação dessas letras.

2.5 Letras Caligráficas

O TeX fornece 26 letras caligráficas maiúsculas $\mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z}$, *também conhecidas como letras de escripte. Elas são produzidas por um tipo de estilo especial invocada com o comando \mathcal.*

```
Escolha $\mathcal{F}$ tal que ...
```

Escolha $\mathcal{F}$ *tal que...*

2.6 Alguns Símbolos Matemáticos

TEX produz uma infinidade de símbolos matemáticos. Alguns deles como $+$ *e* $>$ *, são produzidos usando o teclado. Outros são obtidos como nas tabelas de 2.2 até 2.5. Os símbolos sombreados requerem o pacote latexsym que é incluído no sistema pelo comando* $\backslash usepackage\{latexsym\}$*. Símbolos adicionais podem ser obtidos colocando um símbolo sobre um outro com o comando* $\backslash stackrel$ *ou pelo ambiente* $\backslash array$*, apresentado mais adiante. É possível também colocar uma barra invertida para à esquerda,* $\backslash$*, num símbolo digitando* $\backslash not$ *antes dele.*

```
Se $x \not <,$  então \(x \not \leq y-1. \)
```

Se $x \not<$*, então* $x \not\leq y-1$.

Existem alguns símbolos matemáticos cujo tamanho depende da espécie de ambiente em que ele aparece. Eles são maiores nos ambientes $\backslash displaymath$ *e* $\backslash equation$ *que nos outros ambientes. Esses símbolos estão listados na tabela 2.6, onde são apresentados ambas as versões, grandes e pequenas.*

$\pm$ \pm	$\cap$ \cap	$\diamond$ \diamond	$\oplus$ \oplus
$\mp$ \mp	$\cup$ \cup	$\triangle$ \bigtriangleup	$\ominus$ \ominus
$\times$ \times	$\uplus$ \uplus	$\bigtriangledown$ \bigtriangledown	$\otimes$ \otimes
$\div$ \div	$\sqcap$ \sqcap	$\triangleleft$ \triangleleft	$\oslash$ \oslash
$*$ \ast	$\sqcup$ \sqcup	$\triangleright$ \triangleright	$\odot$ \odot
$\star$ \star	$\vee$ \vee	$\lhd$ \lhd	$\bigcirc$ \bigcirc
$\circ$ \circ	$\wedge$ \wedge	$\rhd$ \rhd	$\dagger$ \dagger
$\bullet$ \bullet	$\setminus$ \setminus	$\unlhd$ \unlhd	$\ddagger$ \ddagger
$\cdot$ \cdot	$\wr$ \wr	$\unrhd$ \unrhd	$\amalg$ \amalg

Tabela 2.2: Tabela de símbolos de operações binárias(para sombrá-las colocar o pacote latexsym)

```
Aqui mostramos como alguns símbolos aparecem:

\[ \sum_{i=1}^{i} = \int_{0}^{1} f \]

e no texto:

\( \sum_{i=1}^{n} x_{i} = \int_{0}^{i} f \)
```

$\leq$ \leq	$\geq$ \geq	$\equiv$ \equiv	$\models$ \models
$\prec$ \prec	$\succ$ \succ	$\sim$ \sim	$\perp$ \perp
$\preceq$ \preceq	$\succeq$ \succeq	$\simeq$ \simeq	$\mid$ \mid
$\ll$ \ll	$\gg$ \gg	$\asymp$ \asymp	$\parallel$ \parallel
$\subset$ \subset	$\supset$ \supset	$\approx$ \approx	$\bowtie$ \bowtie
$\subseteq$ \subseteq	$\sqsupseteq$ \sqsupseteq	$\doteq$ \doteq	$\frown$ \frown
$\in$ \in	$\ni$ \ni	$\notin$ \notin	\dagger
$\bullet$ \bullet	$\setminus$ \setminus	$\unlhd$ \unlhd	$\ddagger$ \ddagger
$\cdot$ \cdot	$\wr$ \wr	$\unrhd$ \unrhd	$\amalg$ \amalg

Tabela 2.3: Símbolos de relações(para sombreá-los colocar o pacote latexsym)

$\leftarrow$ \leftarrow	$\Longleftarrow$ \Longleftarrow	$\uparrow$ \uparrow
$\Leftarrow$ \Leftarrow	$\Longleftarrow$ \Longleftarrow	$\Uparrow$ \Uparrow
$\rightarrow$ \rightarrow	$\longrightarrow$ \longrightarrow	$\downarrow$ \downarrow
$\Rightarrow$ \Rightarrow	$\Longrightarrow$ \Longrightarrow	$\Downarrow$ \Downarrow
$\leftrightarrow$ \leftrightarrow	$\longleftrightarrow$ \longleftrightarrow	$\updownarrow$ \updownarrow
$\mapsto$ \mapsto	$\longmapsto$ \longmapsto	$\nearrow$ \nearrow
$\hookleftarrow$ \hookleftarrow	$\hookrightarrow$ \hookrightarrow	$\searrow$ \searrow
$\leftharpoonup$ \leftharpoonup	$\rightharpoonup$ \rightharpoonup	$\nwarrow$ \nwarrow
$\leftharpoondown$ \leftharpoondown	$\rightharpoonup$ \rightharpoonup	$\nwarrow$ \nwarrow
$\rightleftharpoons$ \rightleftharpoons	$\leadsto$ \leadsto	
$\leftharpoondown$ \leftharpoondown	$\rightharpoondown$ \rightharpoondown	$\swarrow$ \swarrow

Tabela 2.4: Símbolos de setas(para sombreá-los colocar o pacote latexsym)

Aqui mostramos como alguns símbolos aparecem:

$$\sum_{i=1}^{i} = \int_0^1 f$$

e no texto:
$\sum_{i=1}^{n} x_i = \int_0^i f$

2.7 Funções como log

Numa fórmula como $log(x+y)$, o "log", que representa a função logarítmica, é representada por uma única palavra, geralmente com tipos romanos. Entretanto, digitar log numa fórmula denota o produto de três quantidades, l, o e g que são impressas como log. A função logaritmo é denotada pelo comando \log.

```
\( \log xy = \log x + \log y . \)
```

$\aleph$ \aleph	$\prime$ \prime	$\forall$ \forall	∞ \infty
$\hbar$ \hbar	$\emptyset$ \emptyset	$\exists$ \exists	$\square$ \Box
$\imath$ \imath	∇ \nabla	$\neg$ \neg	$\Diamond$ \Diamond
$\jmath$ \jmath	$\surd$ \surd	$\flat$ \flat	$\triangle$ \triangle
ℓ \ell	$\top$ \top	$\natural$ \natural	$\clubsuit$ \clubsuit
$\wp$ \wp	$\bot$ \bot	$\sharp$ \sharp	$\diamondsuit$ \diamondsuit
$\vdash$ \vdash	$\dashv$ \dashv		

Tabela 2.5: Micelâneas de símbolos(para sombreá-los colocar o pacote latexsym)

$\sum$ \sum	$\bigcap$ \bigcap	$\bigodot$ \bigodot
$\prod$ \prod	$\bigcup$ \bigcup	$\bigotimes$ \bigotimes
$\coprod$ \coprod	$\bigsqcup$ \bigsqcup	$\bigoplus$ \bigoplus
$\int$ \int	$\bigvee$ \bigvee	$\biguplus$ \biguplus
$\oint$ \oint	$\bigwedge$ \bigwedge	

Tabela 2.6: Símbolos que variam de tamanho(para sombreá-los colocar o pacote latexsym)

$\log xy = \log x + \log y$.

Outros comandos, para apresentar funções, como \log são apresentadas na tabela 2.7. Dois comandos adicionais produzem a função "mod"(módulo): \bmod, para a relação binária e \pmod para uma expressão com parênteses.

\cos	\csc	\exp	\ker	\limsup	\min	\sinh
\cosh	\deg	\gcd	\lg	\ln	\Pr	\sup
\cot	\det	\hom	\lim	\log	\sec	\tan
\coth	\dim	\inf	\liminf	\max\	\sin	\tanh

Tabela 2.7: Funções como log

2.8 Vetores

2.8.1 O Ambiente array

As matrizes são produzidas com o ambiente array. Este comando tem um único argumento que especifica o número de colunas e linhas de ítens dentro da coluna. Para cada coluna na matriz, existe uma única letra no argumento que espcifica como os ítens na coluna devem ser posicionados: c para centrado, l para esquerdo e r para direito. Dentro do corpo ambiente, as linhas adjacentes são separadas pelo comando \\ e os ítens adjacentes dentro da linha são separados pelo caractere &,

```
\( \begin{array}{clcr}
a + b + c &  uv  & x-y  & 27  \\

a + b     &  u + v    &  z &  137 \\

a  & 3u +  vw & xyz   &  2.987  \\
\end{array} \)
```

$$\begin{array}{clcr} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 137 \\ a & 3u+vw & xyz & 2.987 \end{array}$$

Não deve existir nenhum & após o último ítem numa coluna e nunhum \\ após a última linha. TeX está no modo matemático quando processar cada elemento do vetor, portanto, ele ignora o espaço. Não ponha qualquer extra noo argumento.

2.8.2 Delimitadores

Um delimitador é um símbolo que atua logicamente como um parêntese, com um par de delimitadores fechando uma expressão. A tabela 2.8 lista cada símbolo de TeX visto como um delimitador, juntamente com o comando ou o caracter de entrada que o produz. Esses comandos e os caracteres produzem os delimitadores do tamanho indicado. Entretanto, os delimitadores devem ser bastantes grandes para se adequar a expressão que eles delimitam. Para tornar um delimitador com o tamanho adequado, digite comando \left ou \right antes dele.

```
Delimitadores grandes são geralmente usados com matrizes.

\[ \left( \begin{array}{c}
    \left| \begin{array}{cc}
....\end{array}
\right|
y \\ z
\end{array} \right) \]
```

Delimitadores grandes são geralmente usados com matrizes.

$$\left(\begin{array}{c} \left| \begin{array}{cc} \\ .... \\ \end{array} \right| y \\ z \end{array} \right)$$

Os comandos \left e \right devem vir aos pares, mas os delimitadores que casam não precisam ser os mesmos

```
\[...=\left( \begin{array}{c}
                ... a \\ b
\end{array} \right[ \]
```

(	)	$\uparrow$	\uparrow
{ \{	} \}	$\updownarrow$	\updownarrow
$\lfloor$ \lfloor	$\rfloor$ \rfloor	$\Uparrow$	\Uparrow
$\lceil$ \lceil	$\rceil$ \rceil	$\Downarrow$	\Downarrow
$\langle$ \langle	$\rangle$ \rangle	$\Updownarrow$	\Updownarrow
/ /	\ \		
| |	|| ||		

Tabela 2.8: Delimitadores

$$... = \left(\begin{array}{c} ...a \\ b \end{array} \right[$$

Algumas fórmulas requerem um delimitador grande que não case com nenhum outro ou vice-versa. Os comandos \left e \right devem casar, mas podemos colocar um delimitador invisível digitando um "."após um desses comandos.

```
\[ x = \left\{ \begin{array}{ll}
 y, & \mbox{se $y>0;$} \\
z + y, & \mbox{caso contrário.} \\
                \end{array} \right. \]
```

$$x = \left\{ \begin{array}{ll} y, & \text{se } y > 0; \\ z + y, & \text{caso contrário.} \end{array} \right.$$

2.9 Fórmulas Múltiplas

Os ambientes displaymath e equation constroem fórmulas de uma linha. Uma fórmula é apresentada em várias linhas se ela for uma sequência de fórmulas separadas ou duas fórmulas longas que se adequam numa única linha. Uma sequência de equações ou desigualdades podem ser apresentadas no ambiente eqnarray.

```
...\begin{eqnarray}
x & = & 17y \\
y & > & a + ...+ j +  \\
  &   & k + l + m + n + o + p
\end{eqnarray}
```

...

$$\begin{aligned} x &= 17y && (2.2) \\ y &> a + ... + j + && (2.3) \\ & \quad k + l + m + n + o + p && (2.4) \end{aligned}$$

2.10 Pondo uma Coisa Acima da Outra

Ás vezes, os símbolos numa fórmula são colocados um sobre o outro. O ambiente array é bom para empilhar subfórmulas verticalmente, mas não em peças menores. LaTeX fornece comando especiais para isto e alguns outras espécies de comandos para empilhar símbolos;

2.10.1 Sobre e Sob-linhas

O comando \overline coloca uma linha horizontal sobre seus argumentos.

```
Podemos aninhar sobrelinhas: \( \overline{\overline{x}^{2} + 1 } \)
```

Podemos aninhar sobrelinhas: $\overline{\overline{x}^{2} + 1}$

O comando \underline é análogo ao anterior somente que ele coloca uma linha sob.

```
O valor é $\underline{3x}.$
```

O valor é $\underline{3x}$.

O comando \underline é um comando frágil.

Para colocar chaves horizontais acima e abaixo de uma expressão usamos os comandos \overbrace e \underbrace, respectivamente.

```
\( \overbrace{a + \underbrace{b + c} + d}. \)
```

$\overbrace{a + \underbrace{b + c} + d}$

Na apresentação de uma fórmula, podemos colocar subescrito e superescrito como rótulos nas chaves

```
\[ \underbrace{a + \overbrace{b + \cdots + y}^{24} + z}_{26} \} \]
```

$$\underbrace{a + \overbrace{b + \cdots + y}^{24} + z}_{26}\}$$

2.10.2 Empilhando Símbolos

O comando \stackrel coloca um símbolo emcima de outro.

```
\( A \stackrel{a'}{\rightarrow} B...\)
```

```
\( \vec{x} \stackrel{\mathrm{def}}{=} ... \)
```

$A \stackrel{a'}{\rightarrow} B...$

$\vec{x} \stackrel{\mathrm{def}}{=} ...$

2.11 Espaço no Modo Matemático

No modo matemático o TEX ignora os espaços que forem digitados. Alguns autores gostariam de adicionar mais espaço que aquele fornecido por TEX . Os comandos seguintes adicionam um certo espaço que corresponde um pouco menos que o tamanho da letra x.

```
\,    \:    \;
```

A seguir apresentaremos exemplos de seus usos.

```
\( \sqrt{2} \, x \)
```

$\sqrt{2}x$, *em vez* $\sqrt{2}x$,

```
\( n/ \! \log n  \)
```

que apresenta
$n/\log n$ *em vez de* $n/\log n$.

```
\( \int\!\!\int z\,dx\,dy \)
```

$\int\int z\,dx\,dy$ *m vez de* $\int\int z\,dx\,dy$

2.12 Alterando o Estilo no Modo matemático

2.12.1 Estilos dos Tipos

O LATEX fornece os seguintes comandos para alterar o estilo no modo matemático. Após cada comando vem a sua apresentação.

```
$\mathit{\mbox{itálico} + 2^{t} \Psi log[\psi]}$
```

$itálico + 2^t \blacksquare log[\psi]$

```
$\mathrm{romano + 2^{t} \Psi log[\psi]}$
```

$\mathrm{romano + 2^t \Psi log[\psi]}$

```
$\mathbf{negrito + 2^{t} \Psi log[\psi]}$
```

$\mathbf{negrito + 2^t \blacksquare log[\psi]}$

```
$\mathsf{observe + 2^{t} \Psi log[\psi]}$
```

$\mathsf{observe + 2^t \Psi log[\psi]}$

```
$\mathtt{observe + 2^{t} \Psi log[\psi]}$
```

$\mathtt{observe + 2^t \Psi log[\psi]}$

```
$\mathcal{ \mbox{MAÚSCULAS} \ LETRAS \ SOMENTE}$
```

$\mathcal{C\dashv\updownarrow}$ *MAÚSCULAS* $\mathcal{LETRAS\ SOMENTE}$

2.13 Definindo Comandos e Ambientes

Os arquivos devem ser tão claros quanto permita a estrutura lógica do documento. Qualquer estrutura, como uma notação especial em matemática, que aparece várias vezes no documento deve ser expresso por um comando apropriado ou um ambiente. No que segue apresentaremos esses comandos e ambientes.

2.13.1 Definindo Comandos

O tipo mais simples de estruturas repetidas ocorre quando o mesmo texto aparece em lugares diferentes. A declaração \newcommand define um novo comando para gerar um texto como esse. O primeiro argumento é o nome do comando e o segundo é o texto.

```
 \newcommand{\gn}{$\Gamma_{i}$}
Seja \gn\ o número ... onde \gn\ é ...
...
```

...Seja Γ_i o número... onde Γ_i é...

Este exemplo ilustra um problema muito comum quando definimos comandos para produzir fórmulas matemáticas. O comando \Gamma só pode ser usado no modo matemático, por isso ele está quardado pelo símbolo $. Entretanto, o comando \gn não pode ser usado no modo matemático porque o primeiro $ forçaria TeX deixar o modo matemático e encontraro o \Gamma enquanto no modo de parágrafo. Para resolver este problema TeX fornece o comando \ensuremath, que assegura que o seu argumento está no modo matemático, independente do modo corrente. Portanto,

```
\ensuremath{x^{2}} é equivalente a $x^{2}$
```

se ele aparecer em LR ou nos modos de parágrafos, e é equivalente a

```
x^{2}
```

Com \ensuremath, definimos \gn de modo que ele possa ser usado em qualquer modo.

```
\newcommand{\gi}{\ensuremath{\Gamma_{i}}}

....

Seja \gi\ ... onde $e^{\gi} = \nu(s)$.
```

Γ_i ... onde $e^{\Gamma_i} = \nu(s)$

Esse comando além de tornar o texto de entrada mais legível ele nos permite definir nossos próprios comandos, podendo com isso encomizar muita digitação. Os nomes de comandos e ambientes de LaTeX podem ser usados para abreviar comandos usados muito frequentemente. Por exemplo, a declaração

```
 \newcommand{\be}{\begin{enumerate}}
\newcommand{\ee}{\end{enumerate}}
```

define \be...\ee como sendo equivalente a

```
\begin{enumerate}....\end{enumerate}
```

Para estruturas repetitivas tendo componentes que variam, podemos definir comandos com argumentos dando a \newcommand um argumento opcional.

```
\newcommand{\gno}[2]{ensuremath{\mathit{gnu}(#1;#2)}}
como \gno{5x}{y} e \gno{5x-1}{y+1}...
```

Como $gnu(5x,y)$ *e* $gnu(5x-1;y+1)$*...*

O argumento opcional 2(dentro dos colchetes) especifica que \gnu tem dois argumentos. ♯ e ♯ 2 no último argumento de \newcommand são chamados parâmetros. Eles são substituídos pelo primeiro e segundo argumentos, respectivamente, de \gno, quando o comando for usado. Um comando pode ter até nove argumentos.

Um comando pode ser definido em termos de outro.

```
\newcommand{\usegno}{\gno{5x}{y}}

Esta definição de \gno{0}{1} fornece \usegno\ com o valor esperado.
```

Esta definição de $gno(0;1)$ *fornece* $gnu(5x;y)$ *com o valor esperado.*

Usar \newcommand para definir um comando que já existe produz um erro. A declaração \renewcommand redifine um comando já definido. Ele tem o mesmo argumento que \newcommand. Não redefina um comando existente a menos que você saiba o que está fazendo.

Os comandos \newcommand e \renewcommand são declarações. Seus escopos são determinados pelas regras dadas no capítulo 1. É melhor colocar todas definições de comando juntos no preâmbulo, com comentários descrevendo o que o comando faz.

2.14 Definindo Ambientes

O comando \newenvironment é usado para definir um novo ambiente. Um comando da forma

```
 \newenvironment{coisa}{begin texto}{end texto}
```

define um ambiente coisa no qual TEX substitui um comando \begin{coisa} pelo \begin{texto} e o comando \end{coisa} por \end{texto}. Em geral, um novo ambiente é definido em termos de um ambiente já existente, como \begin{itemize},

```
\newenvironment{emphit}{\begin{itemize}\em}{\end{itemize}}
este é um exemplo de um ambiente definido pelo usuário:
\item Este ambiente produz ítens que são enfatizados.
\end{emphit}
```

Este é um exemplo de um ambiente definido pelo usuário

- **Este ambiente produz ítens que são enfatizados.**

2.15 Teoremas e Semelhantes

Geralmente, os textos matemáticos incluem teoremas e/ou estruturas, como lemas, proposições, axiomas, conjecturas, e assim por diante. Os texto não-matemáticos podem conter estruturas similares: regras, leis, hipóteses, princípios, etc. Ter um ambiente embutido para cada possibilidade facilita a formação de textos. Por isso, TeX fornece a declaração \newtheorem para definir ambientes para estruturas particulares, como teorema.

O comando \newtheorem tem dois argumentos: o primeiro é o nome do ambiente, o segundo é o texto usado para rotulá-lo.

```
\newtheorem{adivinha}{Conjectura}
As conjecturas são enumeradas consecutivamente do início do documento. Este é o quarto:

\begin{adivinha}
Todas as conjecturas são interessantes, mas algumas conjecturas são mais interessantes.
\end{adivinha}
```

As conjecturas são enumerada consecutivamente do início do documento. Esta é a primeira:

Conjectura 1 *Todas as conjecturas são interessantes, mas algumas conjecturas são mais interessantes.*

A declaração \newtheorem pode vir em qualquer lugar do documento, mas é melhor que ele seja colocado no preâmbulo.

O argumento opcional final para \newtheorem tem como efeito que os ambientes como teorema seja numerado dentro da unidade de seção especificada.

```
\newtheorem{axioma}{axioma}[section]
Este é o primeiro axioma da seção 3:
\begin{axioma}
Todos os axiomas são muito aborrecidos.
\end{axioma}
```

Este é o primeiro axioma da seção 3:

Axioma 2.15.1 *Todos os axiomas são muito aborrecidos.*

Os ambientes como teoremas podem ser numerados dentro de qualquer unidade de seção. Usando chapter em vez de de section tem como efeito que os axiomas sejam numerados dentro do capítulo.

Às vezes queremos que diferentes estruturas de teorema compartilhem a mesma sequência de numeração como, por exemplo, que o escólio que segue imediatamente Conjectura 5 seja numerado 6.

```
\newtheorem{adivinha}{Conjectura}
\newtheorem{lema}[adivinha]{Lema}

\begin{adivinha}
Algumas coisas boas...
\end{adivinha}
\begin{lema}
 Não existe nada interessante.
\end{lema}
```

.

Conjectura 2 *Algumas coisas boas...*

Lema 3 *Não existe nada interessante.*

O argumento opcional advinha no segundo comando backslashnewtheoremlema[adivinha]Lema especifica que o ambiente Lema deve ser numerado na mesma sequência que o ambiente adivinha.

O ambiente como teorema definida com \newtheorem tem um argumento opcional que é geralmente usado para o nome do inventor ou o nome comum de um de um teorema, definição ou axioma.

```
\begin{adivinha}[Wiles, 1985]
Existem inteiros $n>2$, $x$,...
\end{adivinha}
```

Conjectura 4 *Existem inteiros ...*

2.16 Figuras e Semelhantes

2.16.1 Figuras e Tabelas

Algo como figura não pode ser quebrada, ela deve "flutuar" em lugares convenientes, como o início da página seguinte. LaTeX fornece dois ambientes para resolver esse problema: figure e table. A captura que aparece na figure ou no table é obtida com o comando \caption, que apresenta o texto em seu argumento.

```
\begin{figure}
Neste espaço está apresentada a figura.
\caption{Esta é a figura para ilustrar...\label{fig1}}
\end{figure}
```

O ambiente \table tem a mesma apresentação de \figure, apenas que será chamada de tabela.

O rótulo \label dentro do comando \caption serve para que possamos nos referir àquela figura ou tabela. Por exemplo,

```
observando a fugura ~\ref{fig1}
```

observando a figura 1.

2.17 Notas Marginais

Uma nota marginal é feita com o comando \marginpar, tendo um texto como seu argumento.

```
... colocando na margem,
\marginpar{\em Esta é uma nota marginal}
```

2.18 Alinhando em Colunas

Os ambientes \tabbing e \tabular alinham os textos em colunas. O ambiente \tabbing nos permmite estabelecer paradas como o tab do teclado, enquanto o ambiente \tabular é análogo ao ambiente array, exceto que ele é para um texto comum e não um ambiente matemático. Os ambientes \tabbing e \tabular diferem nos seguintes pontos.

- *O ambiente \tabbing pode ser usado somente no modo parágrafo e constroi um parágrafo separado. O ambiente \tabular pode ser usado em qualquer modo. Ele pode colocar uma tabela no meio de uma fórmula ou linha.*
- *TeX pode iniciar uma nova página no meio de um ambiente \tabbing, mas não no meio de um ambiente \tabular. Portanto, um ambiente \tabbing longo pode aparecer no texto de execução, mas um \tabular longo deve ser somente em figuras ou tabelas.*

- *TEX automaticamente determina os comprimentos das colunas no ambiente tabular. No ambiente \tabbing nós mesmos temos que fazer isso.*

- *Uma mudança de formnato no meio do ambiente \tabbing é mais fácil do que no \tabular.*

2.18.1 O ambiente Tabbing

No ambiente \tabbing alinhamos um texto em colunas estabelecendo os espaços tab. O tab é obtido com o comando \ = e \ >, enquanto linhas são são separadas pelo comando \\.

```
O ambiente \tabbing inicia uma nova linha.
\begin{tabbing}
 Se \= estiver chovendo, \\
\> então \= ponha as galochas. \\
\> \mbox{Caso contrário,} \> ria. \\
pode sair de casa.
\end{tabbing}
```

O ambiente \tabbing inicia uma nova linha.

Se estiver chovendo,
então ponha as galochas.
Caso contrário, ria.
pode sair de casa.

2.19 O Ambiente Tabular

O ambiente \tabular é similar ao ambiente \array. Ele é diferente do ambiente \array em dois pontos:

- *Ele pode ser usado em qualquer modo e;*

- *seus ítens são processados no modo* LR*, em vez de no modo matemático. Isto torna o ambiente \tabular melhor para tabular listas e \array para melhor para fórmulas matemáticas.*

Nesta seção descreveremos algumas características usadas principalmente com o ambiente \tabular, embora elas se apliquem também ao \array.

Se no ambiente \tabular colocarmos | nos argumentos, TEX coloca uma linha vertical na coluna correspondente. Para colocar uma linha horizontal usamos o comando \line após o comando \\. O comando \cline{i − j} desenha uma linha horizontal da coluna i até a coluna j, inclusive.

```
\begin{tabular}{||l|lr||} \line
gno & gam & $13,65$ \\ \cline{2-3}
    & cada  & ,01 \\ hline
gno & substancial & 92,5 \ \ \cline{1-1} \cline{3-3}
emu &        & 33,33, \\ hline
armadillo  & congelado  & 8,99  \\ hline
\end{tabular}
```

gno	*gam*	*$ 13,65*
emu		*33,33,*
armadillo	*congelado*	*8,99*

2.20 Simulando Texto Digitado

Um documento impresso pode conter texto digitado simulado, por exemplo, o manual de intrunção para um programa de computador normalmente mostra os dígitos usados. A declaração \ttfamily tem como efeito de fazer com que TeX produza o texto num estilo digitalizado, mas não inibe TeX de quebrar o texto em linhas. O ambiente \verbatim nos permmite digitalizar o texto exatamente da maneira que quizermos que ele apareça no documento.

```
  \begin{verbatim}
  ...  como o arquivo de entrada é formatado.  \ begin{verbatim}
O que o ♯ % | & $ ‘está ”indo"{ sobre} aqui \ hoje \\\\ ????  \end{verbatim
  \end{verbatim}
```

Cada espaço digitado produz um espaço na saída, e novas linhas começam exatamente onde digitarmos. No ambiente \verbatim caracteres como \ e \{ são tratados como caracteres ordinário. Realmente, podemos digitar qualquer coisa no corpo de um ambiente \verbatim exceto para o décimo quarto caractere na sequência \end{verbatim}

Capítulo 3

Circulando Informação

Frequentemente temos que mover informação de um lugar para outro. Por exemplo, as informações contidas no conteúdo são obtidas dos comandos de seções que estão espalhadas no arquivo de entrada. Analogamente, o comando LaTeX que gera a referência a uma equação deve obter o número da equação do ambiente $\backslash equation$*, que pode ocorrer várias seções anteriores. Neste capítulo descreveremos as características que fazem com o LaTeX mova informações através do texto.*

O LaTeX precisa de dois passos para mover uma informação: Um deles para encontrar a informação e o outro para colocá-la no texto. (Ás vezes é preciso três passos.) Para compilar o conteúdo, um passo determina o título e as páginas iniciais de todas as seções e o segundo coloca a informação no conteúdo. Em vez de dar dois passos toda vez que executado, LaTeX só lê o arquivo de entrada e salva a informação referenciada num arquivo especial para usar na próxima vez. Por exemplo, se exemplo.tex tivesse um arquivo especial para produzir um conteúdo, LaTeX escreveria a informação necessária no arquivo $exemplo.toc$*. Ele usaria a informação na versão corrente de exemplo.toc para produzir o conteúdo na vez que LaTeX estivesse executando com exemplo.tex como entrada. As informações de referência do LaTeX é, assim, "antiga", pois elas foram obtidas na execução anterior.*

3.1 O Conteúdo

Um comando $\backslash tableofcontent$ *produz um conteúdo. Mais precisamente, ele faz duas coisas:*

- *Leva LaTeX a escrever um novo arquivo, toc, isto é, um arquivo com o mesmo nome que o arquivo de entrada e a extensão toc, com a informação necessária para gerar o conteúdo.*
- *Lê a informação da versão anterior do arquivo toc para produzir o conteúdo com cabeçário.*

Os comandos $\backslash listoffigures$ *e* $\backslash listoftables$ *produzem uma lista de figuras e de tabelas respecivamente. Eles funcionam exatamente como o comando* $\backslash tableofcontent$*,*

exceto que LaTeX escreve um arquivo com extensão lof, quando faz uma lista de tabelas. Às vezes, podemos não estar satisfeitos com o modo como LaTeX imprime um conteúdo ou uma lista de figuras ou tabelas. Em outra oportunidade descreveremos como podemos escrever nosso próprio formato.

3.2 Referências Cruzadas

Uma razão para enumerar objetos como figuras e equações é para poder referenciá-las aos leitores, como em "veja a figura 3, para maiores detalhes". Não queremos que o "3" apareça no arquivo de entrada, pois se precisarmos acrescentar outra figura, ela pode se tornar a Figura 4. O que devemos fazer é colocar uma chave deixando LaTeX traduzir a referência no número da figura. À chave é atribuído um número pelo comando $\backslash label$ *que é impresso pelo comando* ~$\backslash ref$. *O comando* $\backslash label$ *que aparecem nos textos comuns atribui à chave o número da unidade seccional corrente. Aquele que aparece numerado dentro de um ambiente atribui esse número à chave.*

```
A equação ~\ref{eq:euler} na seção ~\ref{sec-inicial} abaixo é o famoso
resultado de Eulwer. \\
\vdots
\subsection{Primeiros resultados}
\label{sec-inicial}
A equação de Euler:
\begin{equation}
e^{i\pi} + 1 = \label{eq:euler}
\end{equation}
combina os cinco mais importantes números numa única equação matemática.
```

A equação 3.1 na seção 3.2.1 abaixo é o famoso resultado de Euler.

$\vdots$

3.2.1 Primeiros resultados

A equação de Euler:

$$e^{i\pi} + 1 = 0 \tag{3.1}$$

combina os cinco mais importantes números numa única equação matemática.

Neste exemplo, o comando $\backslash label\{eq:euler\}$ *atribui a chave* {eq:euler} ao número da equação e o comando ~$\backslash ref\{eq:euler\}$ gera o número de equações. A chave pode ser uma sequência de letras, dígitos ou caracteres de pontuação. As letras minúsculas são diferentes das maiúculas. Além do comando de seção, o ambiente seguinte também gera números que podem ser atribuídos à chave com um comando $\backslash label$: *equation*, *eqnarray*, *enumerate*(atribui o número do ítem corrente), *figure*, *table* e qualquer ambiente parecido com teorema definido com o comando $\backslash newtheorem$.

O comando $\backslash label$ pode, normalmente, aparecer em qualquer lugar usual. Para atribuir um número de uma unidade seccional a uma chave podemos colocar o comando $\backslash label$

em qualquer lugar dentro da unidade, exceto dentro do argummento de um comando ou ambiente, podemos colocá-lo no argumento do comando seccional. Para referencial uma equação particular num ambiente *eqnarray* colocamos o comando $\backslash label$ em qualquer lugar entre o $\backslash\backslash$ ou $\backslash begin\{eqnarray\}$ que começa a equação e $\backslash\backslash$ ou $\backslash end\{eqnarray\}$ que o termina. A posição do comando $\backslash label$ numa figura ou tabela é menos óbvio: ele deve vir apõs o comando $\backslash caption$ ou o seu argumento.

```
A figura ~\ref{fig:novo} mostra a evolução da salamandra de sua origem nos
tempos jurássicos...
\begin{figure}[h]

\caption{O novo em destaque.}\label{fig:novo}
\end{figure}
```

A figura 3.1 mostra a evolução da salamandra de sua origem nos tempos jurássicos...

Figura 3.1: O novo em destaque.

O comando $\backslash caption$ dentro do ambiente de sua *figura* ou *tabela* atua como um comando seccional dentro do documento. Do mesmo modo que um documento tem múltiplas seções, uma *figura* ou *tabela* pode ter múltiplos $\backslash caption$.

O comando $\backslash pageref$ é similar ao comando ~$\backslash ref$, exceto que ele produz o número da página onde o comando $\backslash label$ correspondente aparece.

```
Veja a página ~\pageref{''significado" } para mais detalhes

\vdots

O significadeo \label{''significado" } da vida, do universo e ...
```

Veja a página 33 para mais detalhes

⋮

O significadeo da vida, do universo e ...

Os comandos $\tilde{\backslash} ref$ e $\tilde{\backslash} pageref$ geram somente números e, portanto, devemos digitar a palavra *página* para produzir "página 4".

3.3 Bibliografia e Citações

Uma citação é uma referência cruzada para outra publicação, como um artigo de jornal, chamado a *fonte*. O método moderno de citação de uma fonte é com uma referência cruzada com uma entrada para uma lista da fonte no fim do documento. Com LaTeX a citação é produzida pelo comando $\backslash cite$, tendo a chave da citação como seu argumento.

```
Embora eles tenham desaparecidos da Fiji ~\cite{tom-ix, dick: ens, harry + d}, Knudson ~\cite[page
```

Embora eles tenham desaparecidos da Fiji[4,15,36], Knudson[páginas 22-333] mostrou que...

Uma chave de citação pode ser uma sequência de letras, dígitos ou caracteres de pontuação, exceto que ela não pode conter uma vírgula. No exemplo anterior, LaTeX determinou que a chave da citação *kn:gnus* corresponda ao rótulo fonte 67. Como LaTeX faz isso depende de como produzimos a lista das fontes. A melhor maneira de produzir a lista de fontes é com um programa separado, chamado BibTeX , descrito na próxima seção. Podemos também produzir por nós próprios, como explicaremos numa seção posterior.

3.4 Usando a Biblioteca

BibTeX é um programa separado que produz uma lista de fontes para um documento, obtendo a informação de um banco de dados bibliográfico. Para usar BibTeX devemos incluir em nosso arquivo de entrada um comando *bibliography* cujo argumento especifica um ou mais arquivos que contém o banco de dados. Os nomes dos arquivos do banco de dados devem ter a extensão *bib*. Por exemplo, o comando

```
\bibliografy{insetos,animal}
```

especifica que a lista de fontes é para ser obtida das entradas dos arquivos *insetos.bib* e *animal.bib*. Não deve existir espaço seguindo a vírgula. Mais adiante explicaremos como fazer arquivos de bancos de dados bibliográficos.

BibTeX cria uma lista de fontes contendo as entradas para todas as entradas de citações, especificadas pelo comando *cite*. Os dados para lista de fontes são obtidos do banco de dados bibliográfico, que devem ter uma entrada para cada chave de citação. Um comando *nocite* do arquivo de entrada no LaTeX faz com que a entrada especificada apareça na lista de fontes, mas não produz saída. Por exemplo, $\backslash nocite\{g : nu, g : nat\}$ faz com BibTeX coloque as entradas do banco de dados bibliográfico tendo chave *g:nu* e *g:nat* na lista de fontes. O comando $\backslash nocite*$ faz com todas as entradas no banco de dados bibliográfico sejam colocadas na lista de fontes. O comando *cite* pode estar em qualquer lugar após o comando $\backslash begin\{document\}$, porém ele é frágil.

Para usar BibTeX o nosso arquivo de entrada do LaTeX deve conter um comando *bibliographystyle*. Este comando especifica o estilo bibliográfico, que determina o formato da lista de fontes. Por exemplo, o comando

```
\bibliographystyle{plain}
```

especifica que as entradas devem ser formatadas como especificadas pelo estilo bibliográfico *plain*. O comando *bibliographystyle* pode estar em qualquer lugar após o comando $\backslash begin\{document\}$. Os estilos bibliográficos padrão do LaTeX são:

- *plain*, um formato mais ou menos como sugerido por *Van luene,* onde as entradas são ordenadas alfabeticamente e rotuladas com números.

- *unsrt*, o mesmo que *plain*, exceto que as entradas aparecem na ordem de suas primeiras citações.

- *alpha*, o mesmo que plain, exceto que as fontes são rotuladas como "Knut66", formada com o nome do autor e o ano de publicação.

- *abbrv*, O mesmo que *plain*, exceto que as entradas são compactas, porque os primeiros nomes de meses e nomes de jornais são abreviados.

Existem muitos estilos bibliográficos, inclusive aqueles que produzem listas de fontes no formato usados por vários jornais científicos. A documentação para o programa BIBTEX explica como criamos nosso próprio estilo bibliográfico. A lista de fontes normalmente é formatada naquilo que Van Luene chama *estilo comprimido.* A opção do *document-class* $\backslash openbib$ faz com que ele seja formatado num estilo aberto.

Uma vez tenhamos criado um arquivo de entrada contendo comandos LATEX apropriados, efetuamos a seguinte sequência de etapas para produzir a saída final:

- Executar o LATEX no arquivo de entrada, que assumimos tendo nome *meuArquivo.tex*. O LATEX reclama que todas as citações estão indefinidas, porque não existe ainda uma lista de fontes.

- Executar BIBTEX digitando algo como *bibtex* meuArquivo. Agora, a saída do LATEX conterá a lista de fontes. Entretanto, o LATEX ainda reclamará que nossas citações estão indefinidas, pois a saída produzida pelo comando $\backslash cite$ é baseada na informação obtida da lista de fontes da última vez que LATEX foi executado no arquivo.

- Executar o LATEX mais uma vez o arquivo *meuArquivo*.

Se acrescentarmos ou removermos uma citação teremos que pecorrer todo esse procedimento, novamente, para obter os rótulos da citação e a lista de fontes corretas. No entanto, elas não precism estar corretas até a impressão final.

3.5 Faça Por Si Próprio

Uma lista de fontes é criada com um ambiente *thebibliography,* que é como o ambiente *enumerate*, descrito anteriormente, exceto que:

- Cada ítem começa com um comando $\backslash bibitem$ cujo argumento é a chave da citação.[1]

- O ambiente *thebibliography* tem um argumento que deve ser um fragmento de texto, do mesmo comprimento ou ligeiramente maior que o rótulo do maior ítem na lista de fontes.

[1] Os comandos $\backslash bibitem$ e $\backslash cite$ funcionam parecidos com os comandos $\backslash label$ e ~$\backslash ref$.

3.6 Referências

```
Knudson ~\cite{kn:gnus} mostrou que no Ártico

\vdots

\begin{thebibliography}{99}

\vdots

\bibitem{kn:gnu} D. E. Knudson.
\emph{1996 World Gnus Almnac.}
Permafrost Press, Novosibirisk.
\end{thebibliography}
```

Knudson [**?**] mostrou que no Ártico

⋮

Referências Bibliográficas

⋮

[1] D. E. Knudson. *1996 World Gnus Almnac.* Permafrost Press, Novosibirisk.

3.7 Separando as Entradas

Um documento muito grande precisa de muitas entradas. Em vez de colocar todas as entradas num único arquivo muito grande, pode ser mais interessante separar as entradas em vários arquivos menores. Independente de como os vários arquivos serão usados, existe um que é o arquivo *raíz*. Ele é aquele cujo nome digitamos quando executamos o LaTeX.

O comando $\backslash input$ fornece a maneira mais simples de separar a entrada em vários arquivos. O comando $\backslash input\{kn : gnu\}$, no arquivo raíz, faz com que LaTeX insira o conteúdo do arquivo *gnu.tex* na localização corrente de nosso manuscrito, exatamente como se o comando $\backslash input\{gnu\}$ fosse removido do arquivo raíz e substituído pelo conteúdo do arquivo *gnu.tex*. (Entretanto, o arquivo de entrada não é alterado.) O arquivo *gnu:tex* pode também conter um comando $\backslash input$ chamando outro arquivo que pode ter seus próprios comandos $\backslash input$ e assim por diante. Além de separar as entradas num feixe de arquivos de tamanhos convenientes, o comando $\backslash input$ também facilita o uso da mesma entrada em documentos diferentes. Mesmo não utilizando essa maneira, poderemos desejar reutilizar declarações. Podemos manter um arquivo contendo declarações que serão usados em todos nossos documentos, como definições de comandos e ambientes para nossa própria estrutura lógica. Poderemos, mesmo, começar nosso arquivo com um comando $\backslash input$ e colocar o comando $\backslash documentclass$ em nosso arquivo de declarações.

Uma razão para separar a entrada em arquivos é executar o LaTeX somente em partes do documento de modo que, quando fizermos alterações, somente precisem ser processados aquelas partes que foram alteradas. Por isso deveremos usar o comando $\backslash include$ e não o comando $\backslash input$. Os dois comandos são similares na medida em que $\backslash include\{gnu\}$ também especifica que o conteúdo do arquivo *gnu.tex* deve ser inserido no mesmo lugar. Entretanto, com o comando $\backslash include$ instruímos LaTeX a inserir o arquivo ou omiti-lo e processar todos os textos que o suceder, como se o arquivo tivesse sido inserido, as páginas numeradas, assim como as seções, as equações, etc., como se o texto do arquivo omitido tivesse sido incluído.

Para executar LaTeX somente numa parte do documento o preâmbulo deve conter um comando $\backslash includeonly$, cujo argumento é uma lista de arquivos(somente os primeiros

nomes). O arquivo especificado por um comando *include* só é processado se ele aparecer no argumento do comando *includeonly*. Portanto, se o preâmbulo contiver o comando *includeonly*{*gnu*, *gnat*, *gnash*} um comando *include*{*gnat*} faz com que o arquivo *gnat.tex* seja incluído, enquanto o comando *include*{*rmadlo*} faz com que LaTeX processe o texto que o sucede sob a hipótese de que o arquivo omitido seja exatamente o mesmo como se ele fosse a última vez que tivesse sido incluído.

O arquivo raíz completo é sempre processado. Se o preâmbulo não contiver um comando *includeonly*, cada comando *include* insere seu arquivo. O comando *includeonly*{} (sem argumentos) instrui o LaTeX a omitir todos os arquivos de *include*. Um comando *include* só pode aparecer após o comando *begin*{*document*}.

O comando *include* tem uma característica que limita sua utilidade: o texto incluído sempre começa numa nova página, como o texto seguindo imediatamente o comando *include*. Portanto, ele só funciona corretamente se o texto do *include* e o texto seguinte tiverem que começar numa nova página. Por exemplo, se consistir de um mais capítulos completos. Enquanto estivermos escrevendo um capítulo é bom separá-lo em arquivos menores com os comandos *include*. Quando terminarmos esse capítulo poderemos combinar aqueles arquivos num único arquivo.

Outra dificuldade com o mecanismo de *include* é que alterando o documento pode ser que seja preciso reprocessar alguns arquivos de *include* não alterados para obter corretamente os números das páginas, seções, etc. Quando saltamos um arquivo de *include* os que o sucedem são baseados no texto do arquivo que foi ultimamente processado. Suponha que o arquivo raíz contenha o comando

```
 \include{gnu}
\chapter{Armadillo}
```

então, um comando *includeonly* no preâmbulo faz com que o comando *include* omita o arquivo *gnu.tex*. Se o texto em *gnu* terminou no capítulo 5, página 56, a última vez em que ele foi processado, mesmo que tivéssemos acressentado mais sete capítulos, a sexagéssima página do texto antes do comando *include* desde então, o comando *chapter* produzirá o capítulo 6, começando com a página 67. Em geral, para assegurar que tudo está sendo enumerado corretamente, deveremos reprocessar o arquivo *include* se uma alteração do arquivo do texto precedente alterar o número no texto produzido por aquele arquivo.

Quando escrevemos um documento grande(como um livro), provavelmente devemos fazer cada capítulo num arquivo de *include* separado. Processar cada arquivo separadamente, enquanto estivermos escrevendo-o ou revisnado-o e não nos preocuparmos com números que não estejam apropriadamente casando. Quando terminarmos as alterações, o LaTeX será executado no documento completo. Então a saída final será produzida toda de uma vez, usando o comando *includeonly*.

3.8 Fazendo um Índice ou um Glossário

Existem duas etapas na construção de um índice ou glossário: juntar as informações que vão estar nele e gerar a entrada LaTeX para produzi-lo. A seção ?? descreve a primeira

etapa. A maneira mais fácil de efetuar a segunda etapa será descrita no próximo capítulo. A seção ?? descreve como produzir um índice ou glossário se não usarmos *MakeIndex.*

3.9 Compilando as Entradas

Compilar um índice ou glossário não é fácil, mas LaTeX pode ajudar escrevendo as informações necessárias num arquivo especial. Se o arquivo raíz tiver o nome *meuArquivo*, a informação do índice será escrito no arquivo *"meuArquivo.idx"*. O LaTeX faz um arquivo *idx* se o preâmbulo contiver um comando *\makeindex*. A informação no arquivo é escrita pelos comandos *\index*. O comando *\index{gn}* aparecendo com o texto para a página 42 faz com que LaTeX escreva *\indexentry{gnu}{42}*. no arquivo *idx.* Se não existir o comando *\makeindex* o comando *\index* não faz nada. O pacote *showidx* faz LaTeX imprimir os argumentos de todos os índices na margem. O comando *\index* não produz texto, de modo que digitamos

```
Um gnat \index{gnat} com asas gnadas...
```

para indexar este exemplo de "gnat". É melhor colocar o comando *\index* próximo a palavra que referencia, sem espaços entre eles. É melhor colocar a entrada de índices numa linha separada, como em

```
Quando no curso de um
\index{evento} humano}%
\index{evento,humano}%
evento humano, é preciso  que um povo ...
```

O uso do caractere % foi explicado anteriormente. O procedimento para fazer um glossário é completamente análogo. Em lugar do comando *\index* existe um comando *\makeindex*. O comando *\makeglossary* produz um arquivo com a extensão *glo,* similar ao arquivo *idx,* exceto com a entrada *\glossaryentry* em lugar de *\indexentry.*

O argumento de *\index* ou *\glossary* pode conter qualquer caractere, incluíndo especiais como \ e $. Os comandos *\index* e *\glossary* são frágeis. Além disso, esses comandos não devem aparecer no argumento de qualquer outro comando, se seu próprio argumento contiver qualquer um dos dez caracteres especiais do LaTeX.

3.10 Produzindo um Índice ou Glossário por si Próprio

Se não usarmos o programa *MakeIndix* poderemos usar o ambiente *theindex* para produzir um índice em formato duas colunas. Cada entrada de índice principal com um comando *\item*. Uma subentrada começa com *\subitem* e uma subsubentrada começa com *\subsubitem*. As linhas em branco entre as entradas são ignoradas. Um espaço vertical extra é produzido pelo comando *\indexspace*, que usualmente é colocado na primeira entrada que começa com uma nova letra.

```
\begin{theindex}
\item gnast 13, 97

\item gnus 24, 37, 233

\item subitem ruim, 39, 236
  \subsubitem muito, 235
\subitem bom, 38, 234
\indexspace

\item armadillo 99, 144
\end{theindex}
```

Índice Remissivo

gnast 13, 97
gnus 24, 37, 233
subitem ruim, 39, 236
    muito, 235
  bom, 38, 234

armadillo 99, 144

Capítulo 4

Outros tipos de Documentos

4.1 Livros

Os livros são diferentes dos artigos(reports), particularmente na sua primeira apresentação e no seu final. A frente de um livro normalmente inclui um título de meia página, uma página para o título principal , um prefácio e uma tabela de conteúdo. Pode também conter reconhecimentos, dedicatórias, uma lista de figuras, uma lista de outros livros na mesma série, etc. A parte de trás, usualmente, inclui um índice e pode conter um pósfácio, reconhecimento, uma bibliografia, etc.

O *documentclass book* não pretende fornecer comandos para todas as estruturas lógicas que possa aparecer na parte de frente e na parte de trás de um livro. É exatamente a mesma que a classe *article* ou *report*, exceto pelo que descreveremos abaixo. Editores individuais podem possuir pacotes com comandos adicionais para estrutura particular que eles desejam em seus livros.

A parte de frente, a principal parte, e a parte de trás começam com três comandos $\backslash frontmatter$, $\backslash mainmatter$ e $\backslash backmattermatter$, respectivamente. Na classe *book* padrão, as páginas da frente são numeradas com numerais romanos. A parte principal e a parte de trás são numeradas com numerais arábicos. Na parte da frente e de trás o comando $\backslash chapter$ não produz um número de capítulo, mas ele faz a entrada da tabela de conteúdo. Portanto, podemos começar o prefácio com

```
\begin{preface}
```

Somente a forma estrela de outros comandos de seção deve ser usada na parte da frente e na parte de trás. A impressão nos dois lados é a normal para livros. Portanto, a opção *twoside* é a normal(automática) na classe *book.* Outra opção normal é *openright,* que faz com que LaTeX comece numa nova página da direita. A opção contrária *openany*, é o normal(automática) para a classe report.

Um livro pode ter mais que uma página de título, cada uma contendo várias informações, além do título e do autor. Por exemplo, o editor, a série e o ilustrador. O comando $\backslash maketitle$ é, portanto, de pouco uso num livro.

4.2 Transparências

O tipo de documento *slide* é usado para fazer transparências. As transparências, normalmente, são impressas ou fotocopiadas para projeção sobre uma tela. Para obter boas transparências é preciso uma informação visual, o que significa que LaTeX não é bem adequado para a tarefa. A seguir apresentaremos algumas razões porque de qualquer maneira queremos usar LaTeX.

- Nossas transparências são baseados no material de um documento LaTeX.
- Nossas transparências têm muitas fórmulas matemáticas.
- Não fazemos transparências com tanta frequência.

As fontes normais(automáticas) da classe de documento *slide* foi especialmente projetado para facilitar a leitura numa certa distância. Parece que não existe outro sistema que faça transparências que sejam legíveis assim. As fontes da classe de documento *slide* são muito maiores do que aquelas usuais. O comando *normalsize* produz mais ou menos o mesmo tamanho de caraacteres que o comando *large* em outra classe de documento. Além disso, a família romana é similar à família *sans serif.*

4.3 Transparências e Capas

Uma transparência particular é produzida com o ambiente:

```
\begin{slide}
\begin{center}
GNATS E GNUS
\end{center}

Os Gnats são muito pequenos e tem um temperamento ruim.

Os Gnus são grandes.
\end{slide}
```

Não podemos executar esse progarma neste texto porque ele só pode funcionar no estilo *documentclass{slide}.*

O texto que aparece na transparência é produzido com os comandos LaTeX comuns. Qualquer comando que fizer sentido para transparências pode ser usado. Dentre os comandos que não faz sentido estão os comandos de seção, os ambientes *figure* e *table* e os comandos para quebrar páginas. Esse último não faz sentido para transparência porque cada transparência deve se ajustar numa única página. Para fazer transparências coloridas devemos usar o pacote *color*, descrito noutro local.

O ambiente *overlay* é usado para uma capa, uma transparência para colocar sobre as outras. Ele é o mesmo que o ambiente *slide,* exceto pela numeração. A primeira

capa seguindo a transparência número 3 é numerado “3-a", o segundo é numerado “3-b"e assim por diante. A melhor maneira de conseguir que um texto no slide e na capa case apropriadamente é ter todo o texto em ambos e colorindo de branco aquele texto que não deve aparecer.

```
\begin{slide}
 \begin{center}
Os Gnats são muito pequenos

e tem um temperamento ruim.

\textcolor{Os Gnus são grandes}
 \end{center}

\end{slide}


\begin{center}
 \textcolor{white}{Os Gnats} e os Gnus
\end{center}
\textcolor{white}{Os gnats são muito pequenos
e tem um temperamento ruim.}

Os Gnus são grandes.
\end{overlay}
```

4.4 Observações

Quando damos uma palestra, ajuda colocar observações para nós próprios para nos lembrar do que dizer. O ambiente *note* produz uma observação de uma página. Observe que seguindo a transparência número 3 estão numerados “3-1” e “3-2”, etc. Podemos imprimir as transparências e as notas separadamente, usando os comandos *onlyslide* e *onlynotes*, descritos abaixo.

É de bom tom planejar uma palestra cuidadosamente e não ultrapassar o tempo. A opção *clock* da classe de documentos *slide* ajuda-nos a seguir o tempo durante nossa palestra. Exatamente antes ou após cada transparência, coloque um comando *addtime*, fornecendo o número de segundos que serão gastos naquela transparência. A quantidade total de tempo(em minutos) que deveria tomar deve ser impresso na parte de baixo de cada observação.

```
\documentclass[clock]{slide}
...
```

```
\addtime{180} % Transparência 1: 3 minutos.
...
\addtime{150} % Transparência 2: 2-1/2 minutos.
 ...
\addtime{120} % Transparência 3: 2 minutos.
...

\begin{note}

Não esquecer de matar aqueles Gnats.
\end{note}
```

4.4.1 Imprimir Somente Algumas Transparências e Notas

Para fazer correções é conveniente gerar somente algumas transparências e as observações de arquivos. O comando $\backslash onlyslide\{4, 7-13, 23\}$ no preâmbulo faz com que LaTeX gere somente as transparências numeradas 4, 7-13(inclusive) e 23, mais todas as outras capas. Os números das transparências no argumento devem estar na ordem ascendente, podendo incluir transparências não existentes. Por exemplo, podemos digitar $\backslash onlyslide\{10-9999\}$ para produzir todas as transparências a partir da décima. O argumento do comando $\backslash onlyslide$ deve ser não vazio.

Existe também um comando análogo $\backslash onlynotes$ para gerar um subconjunto das observações. A numeração das observações 11-1, 11-2, etc, serão todas geradas especificando a página 11 nos argumentos dos comandos $\backslash onlyslides$ e $\backslash onlynotes$. Se o arquivo de entrada tiver um comando $\backslash onlyslides$, mas não o comando $\backslash onlynotes$, então não será produzido nenhuma observação e vice-versa. Incluídos ambos esses comandos o efeito é o esperado, que é o de produzir somente as transperências e as observações especificas.

4.5 Cartas

Para fazer cartas, aquelas que colocamos no envelope e no correio, LaTeX fornece o tipo de documento de carta. Para usar essa classe, começamos nosso arquivo de entrada com um comando $\backslash documentclass$, tendo *letter* como seu argumento principal(não opcional). Podemos fazer qualquer número de cartas com um único arquivo de entrada. O nome e o endereço do remetente, que provavelmente tem as mesmas letras que o da carta toda, são especificadas por declarações. O endereço do remetente é declarado por um comando $\backslash address$, com múltiplas linhas de saída separadas pelos comandos $\backslash\backslash$.

```
\address{1234 Av.. \dos Armadillos \\
Gnu York, GY. 56789}
```

O comando $\backslash signature$ declara o nome do remetente, que aparece no fim da missiva, com o comando de separação de linhas múltiplas, $\backslash\backslash$.

```
\signature{R. (Ma) Dillo \\ diretor de cozinha}
```

Essas declarações são normalmente colocadas no preâmbulo, mas elas são declarações comuns com as regras de escopo normal, podendo aparecer em qualquer lugar no documento.

Cada carta é produzido por um ambiente *carta* separado, tendo o nome e o endereço do destinatário como argumentos. A carta, propriamente, começa com um comando *opening* que gera a saudação.

```
\begin{letter}{Dr..\ G. Nataniel Picard \\
 Exterminadores Acme \\ 33 swat Street \\
Hometown, Illinois 62301}
\opening{Prezado Nataniel Picard,}

estou procupado que o problema do armadillo ainda esteja conosco. Fiz tudo...
```

O endereço do remetente é determinado pela declaração *address*. O LaTeX fornece a data. Um comando *address* e/ou *signature* que se aplicam a uma carta podem ser colocados entre o comando *begin*{*letter*} e o*opening*.

O corpo principal da carta é um LaTeX comum, mas os comandos como *section* que não fazem sentido numa carta não podem ser usados. As cartas teminam com um comando *closing*.

```
... e espero que poderemos nos livrar...

\closing{Até mais ver,}
```

O nome vem na declaração *signature*. O comando *cc* pode ser usado após fechar com a lista de nomes para quem estamos enviando as cópias.

```
 \cc{Barack Obama \\ Lula da Silva}
```

O texto adicional após o comando *closing* deve ser precedido pelo comando *ps*. Este comando não gera texto, mas, apenas “P.S."Podemos alterar a data de nossa carta, usando *renewcommand* para redefinir *today*. Colocamos *renewcommand* no prâmbulo para alterar a data de todas as cartas. Colocamos exatamente antes do comando *opening* para alterar a data da última carta.

```
\begin{letter}{Mr. \ A. Dop \\
Reinaldo dos Gnats}

\renewcommand{\today{Thursay, 12345 B. C.}

\opening{Prezado Mr. \Oop:}

No seu último depoimento para a família Gnats,...
```

Um comando *makelabels* no preâmbulo faz com que LaTeX imprima uma lista de rótulos de mensagens para cada ambiente carta, num formato adequado para cópia xerográfica.

Referências Bibliográficas