

# **Princípios do LaTeX**

Lucas Mafaldo Oliveira

# Princípios do LaTeX

Lucas Mafaldo Oliveira

This book is for sale at <http://leanpub.com/lutex>

This version was published on 2013-10-15



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

©2013 Lucas Mafaldo Oliveira

# Conteúdo

<b>Introdução</b> . . . . .	1
O LaTeX e as humanidades . . . . .	1
O objetivo deste livro . . . . .	1
A importância das ferramentas . . . . .	2
Porque o LaTeX . . . . .	3
Visão geral do livro . . . . .	4
<b>Capítulo 1: Os princípios do LaTeX</b> . . . . .	6
O LaTeX é uma linguagem e não um programa . . . . .	6
Porque separar o conteúdo da formatação . . . . .	7
Como trabalhar com LaTeX . . . . .	10
Escrevendo documentos em LaTeX . . . . .	11
Lendo documentos em LaTeX . . . . .	13
Notas finais . . . . .	14
<b>Capítulo 2: Os primeiros passos</b> . . . . .	16
Instalando o LaTeX . . . . .	16
Escolhendo um editor . . . . .	17
O problema dos caracteres especiais (ou: só unicode salva) . . . . .	17
A solução permanente para os acentos e caracteres estrangeiros . . . . .	20
Notas finais . . . . .	21
<b>Capítulo 3: Os elementos básicos de um documento em LaTeX</b> . . . . .	22
Introdução . . . . .	22
Pensando como o LaTeX . . . . .	23
Os elementos básicos que compõe um documento em LaTeX . . . . .	24
Introdução aos comandos ( <code>\comando{variável}</code> ) . . . . .	24
A estrutura do documento: o conceito de preâmbulo . . . . .	26
A estrutura do documento: o texto . . . . .	28
Falhando no LaTeX: lidando com erros . . . . .	29
Vencendo no LaTeX: o primeiro arquivo . . . . .	31
Os próximos passos . . . . .	33
<b>Capítulo 4: Como o LaTeX lida com o texto</b> . . . . .	35
Introdução . . . . .	35
O conceito de linha no LaTeX . . . . .	35
Como o LaTeX interpreta as linhas do seu documento . . . . .	39
A importância das linhas na solução de erros . . . . .	46

## CONTEÚDO

Como o LaTeX interpreta os espaços . . . . .	46
O problema das aspas . . . . .	47
Divisão silábica no LaTeX . . . . .	51
Os sinais gráficos problemáticos . . . . .	59
Notas finais . . . . .	60
<b>Capítulo 5: Os comandos do LaTeX . . . . .</b>	<b>62</b>
Introdução . . . . .	62
Características gerais dos comandos em LaTeX . . . . .	63
Os comandos do preâmbulo: definindo a classe do documento . . . . .	63
Os comandos do preâmbulo: adicionando pacotes ao seu documento . . . . .	66
Os comandos do preâmbulo: definindo o estilo do documento . . . . .	69
Os comandos do preâmbulo: Adicionando informações sobre o documento . . . . .	70
O conteúdo do documento: criando a estrutura do texto . . . . .	72
O conteúdo do documento: gerando um sumário automaticamente . . . . .	79
O conteúdo do documento: os comandos internos ao texto . . . . .	87
Bônus: inserindo documentos dentro de outros documentos . . . . .	98
Notas finais . . . . .	101
<b>Capítulo 6: Referências bibliográficas no LaTeX . . . . .</b>	<b>103</b>
Introdução . . . . .	103
O LaTeX e as referências bibliográficas . . . . .	104
A delicada combinação entre LaTeX e BibTeX . . . . .	105
O conceito de base de dados bibliográficos . . . . .	106
Como utilizar o banco de dados bibliográficos . . . . .	109
Escolhendo um estilo bibliográfico . . . . .	110
Pacotes bibliográficos: utilizando o natbib . . . . .	111
O pacote bibliográfico do abnTeX2 . . . . .	118
Notas finais . . . . .	122
<b>Capítulo 7: Criando apresentações com o LaTeX . . . . .</b>	<b>124</b>
Introdução . . . . .	124
Criando o documento final de uma apresentação . . . . .	125
O preâmbulo de uma apresentação . . . . .	126
A estrutura de uma apresentação . . . . .	128
O corpo da apresentação . . . . .	129
O modelo de uma apresentação . . . . .	130
Analizando o resultado final . . . . .	132
Notas finais . . . . .	141
<b>Considerações finais . . . . .</b>	<b>142</b>
<b>ANEXOS . . . . .</b>	<b>144</b>
lutex01.tex . . . . .	144
lutex02.tex . . . . .	144
lutex03.tex . . . . .	145
lutex04.tex . . . . .	145

## CONTEÚDO

lutex05.tex . . . . .	146
lutex06.tex . . . . .	146
lutex07a.tex . . . . .	146
lutex07b.tex . . . . .	147
lutex07c.tex . . . . .	147
lutex07d.tex . . . . .	148
lutex07e.tex . . . . .	148
lutex08.tex . . . . .	148
lutex09a.tex . . . . .	149
lutex09b.tex . . . . .	149
lutex10a.tex . . . . .	150
lutex11a.tex . . . . .	150
lutex10b.tex . . . . .	151
lutex11b.tex . . . . .	152
lutex12.tex . . . . .	153
lutex13.tex . . . . .	153
lutex14.tex . . . . .	154
lutex15a.tex . . . . .	154
lutex15b.tex . . . . .	154
bibliografia.bib . . . . .	155
lutex16a.tex . . . . .	155
lutex16b.tex . . . . .	156
lutex16c.tex . . . . .	156
lutex16d.tex . . . . .	157
lutex16e.tex . . . . .	157
lutex17a.tex . . . . .	158
lutex17b.tex . . . . .	158
lutax18a.tex . . . . .	159
lutex18b.tex . . . . .	160

# Introdução

## O LaTeX e as humanidades

Se o leitor resolveu abrir esse livro, é porque provavelmente já ouviu falar sobre o LaTeX. Infelizmente, é igualmente provável que tenha ouvido falar que o LaTeX é uma ferramenta para matemáticos e programadores. A associação está longe de ser arbitrária, mas impede que os pesquisadores das ciências humanas e das humanidades descubram uma ferramenta que tornaria seu trabalho incrivelmente mais produtivo.

A associação entre LaTeX e as ciências da computação remonta ao seu nascimento. Afinal, o sistema TeX original foi desenvolvido por um dos gigantes da área – Donald Knuth – que desejava um sistema tipográfico capaz de transcrever corretamente equações matemáticas. No entanto, em suas mais de três décadas de existência, o TeX adquiriu uma infinidade de novas funcionalidades que o tornaram extremamente úteis mesmo para quem não precisa utilizar nenhuma equação em seus trabalhos.

Dito isto, é verdade que essa herança matemática afasta parte dos pesquisadores de formação humanística. O LaTeX realmente exige alguns conhecimentos de informática acima da média, já que força o usuário a lidar com alguns processos que outros editores deixam por trás da cortina de uma interface visual agradável. Além disso, como o LaTeX é geralmente utilizado por pesquisadores da área de exatas, os guias para o LaTeX também são produzidos por esses pesquisadores para seus futuros colegas. Isso torna esses guias difíceis de compreender tanto pelo aquilo que eles *dizem* como pelo que eles *não dizem*. Por um lado, os leitores das humanidades se assustam com a quantidade de referência à matemática; por outro lado, esses guias deixam implícitos uma série de princípios e informações que são evidentes para o pessoal de exatas, mas que passam desapercebidos pelo pessoal de humanas. Esse impasse, somado ao fato de que o LaTeX realmente exige um investimento inicial considerável em termos de atenção e esforço, afasta os pesquisadores de humanas, o que gera um ciclo que se auto-alimenta: como há poucos pesquisadores de humanas utilizando o LaTeX, os novos alunos acham que o sistema não é interessante para eles e se forma uma nova geração de pesquisadores que desconhecem o LaTeX.

Para ajudar a quebrar esse ciclo, escrevi um pequeno guia para convencer os amigos mais próximos a utilizar o LaTeX no campo da filosofia em geral e da filosofia antiga em particular. Quando vi que o guia estava crescendo, percebi que poderia organizá-lo e transformá-lo em livro. Espero, desse modo, contribuir – um pouco que seja – para incentivar uma adoção mais ampla do LaTeX pela comunidade acadêmica – creio que todos temos muito a ganhar se isso ocorrer.

## O objetivo deste livro

O leitor certamente encontrará uma infinidade de manuais sobre LaTeX na internet. No entanto, além destes manuais serem geralmente voltados para o público de exatas, noto que o foco normalmente é apresentar informações de um modo exaustivo. Isso torna esses guias excelentes

recursos enquanto manual de referência, mas pouco interessantes enquanto material didático para iniciantes. Como os professores bem o sabem, o aprendizado envolve uma dose considerável de repetições, discussões e de esforço em enxergar o mesmo assunto por vários ângulos.

Tendo isso em mente, tentei fazer um livro que fosse mais didático do que exaustivo. O objetivo não é apresentar *tudo* sobre o LaTeX – o que, em todo caso, seria simplesmente impossível – mas levar o leitor que nunca tinha ouvido falar sobre esse sistema ao ponto onde poderá começar a utilizá-lo de modo consciente. Ao final do livro, espero que o leitor seja capaz de produzir textos básicos e – o que é ainda mais importante – conhecer os princípios teóricos que irão lhe permitir interpretar corretamente os guias mais avançados.

## A importância das ferramentas

Em certo sentido, esse livro é um efeito colateral da minha tese de doutorado. Embora o objeto formal da tese não tenha nenhuma relação com o campo tecnológico, o trabalho de *escrever* a tese me fez perceber a necessidade de conhecer melhor as ferramentas do trabalho acadêmico. Enquanto escrevia a tese, fui testando diferentes sistemas e programas até me convencer que o LaTeX era a ferramenta ideal para meu trabalho – e não apenas para escrever a tese, mas também para escrever artigos e preparar apresentações.

Ao longo desses experimentos, percebi mais uma das várias pequenas diferenças culturais que separam os pesquisadores das chamadas *humanidades* daqueles que trabalham no campo das *hard sciences*. O pessoal “das exatas”, desde a graduação, estava sempre discutindo os instrumentos técnicos que utilizavam – inclusive, com frequente e acaloradas discussões sobre os méritos relativos de editores de textos e sistemas operacionais. Enquanto isso, nós, “das humanas”, tendíamos a utilizar o que estivesse à mão – fosse aquilo já utilizado pelos colegas mais próximos ou o que já estivesse previamente instalado no computador recém-adquirido. Parte dessa diferença, evidentemente, se justifica por uma diferença de competência técnica – simplesmente não é preciso ter tanto conhecimento sobre *software* para escrever sobre filosofia, história ou literatura. No entanto, parte dessa diferença também se deve à impressão de que os verdadeiros intelectuais, os *pensadores puros*, devem estar tão profundamente imerso em reflexões teóricas que não lhes resta tempo para questões meramente utilitárias.

Atualmente – e principalmente depois de todo o trabalho prático envolvido em produzir uma tese de doutorado – penso que essa postura está equivocada. Evidentemente, não coloco em discussão o fato de que os problemas científicos e filosóficos devem estar no centro da atenção do pesquisador. A pesquisa deve vir sempre em primeiro lugar. No entanto, justamente por causa da sua importância, não devemos menosprezar as ferramentas técnicas. Não devemos pensar nisso em termos excludentes: *justamente* porque os problemas intelectuais são importantes, devemos escolher instrumentos que tornem nosso trabalho mais eficiente para podermos nos concentrar melhor sobre eles.

Essa questão se torna ainda mais crítica devido à configuração atual da carreira acadêmica. Somos constantemente chamados a produzir artigos e relatórios, a preparar apresentações para aulas e congressos. Passamos, enfim, cada vez mais tempo diante do computador do que diante de livros impressos. Quanto mais eficiente formos na utilização desses instrumentos, mais tempo teremos para nos dedicar à parte crucial do trabalho.

## Porque o LaTeX

Embora isso talvez pareça óbvio, creio que é muito tentador ficarmos presos em uma espécie de *ciclo de subinvestimento técnico*. Os programas que costumamos utilizar possuem uma falsa aparência de simplicidade. Programas como *Word* e *Power Point* foram produzidos tendo em mente as necessidades básicas do usuário médio. Por isso, as funções mais comuns do programa são bem evidentes: basta olhar para a tela para descobrir como mudar a fonte, deixar o texto em negrito, imprimir, etc. Por isso, esses programas exigissem pouquíssimo esforço para fazer coisas simples: em poucos minutos, qualquer pessoa consegue produzir um documento básico.

Os problemas surgem quando suas necessidades começam a fugir daquelas do usuário médio. No momento em que você precisa de recursos mais avançados, tudo fica progressivamente muito mais complicado. Essas funções existem, mas elas precisam ficar escondidas em vários *menus* e *submenus* para dar espaço às ferramentas padrões. Isso lhe coloca em uma posição diametralmente oposta à versão inicial: agora você irá levar várias horas para descobrir como corrigir a formatação das notas de rodapé, a inserção de referências, a formatação do índice, do sumário, etc.

Essas dificuldades aumentam ainda mais devido a um fator específico: os trabalhos acadêmicas normalmente exigem regras muito rígidas de formatação. E não é apenas uma questão da ABNT: praticamente todos os periódicos e editoras internacionais possuem normas particulares bastante específicas. Essas normas dizem respeito tanto à apresentação visual – margens, fontes, espaçamento – quanto às normas internas ao texto, como citações e referências bibliográficas. Infelizmente, os editores convencionais (seja ele o *Word* ou o *Open Office* – o problema não é uma empresa em particular, mas o *estilo* do programa) *não* foram criados com o foco na apresentação tipográfica final, mas apenas em tornar mais fácil a produção de um documento básico. Por isso, esse tipo de programa torna extremamente fácil o ato de *digitar* o texto, mas não possui como competência características garantir a exatidão da apresentação final.

É aqui que entra o LaTeX: ele é uma ferramenta especificamente desenvolvida para as nossas necessidades. Aquilo que é difícil no Word – sumário, índice, bibliografia – se torna ridicamente fácil no LaTeX. No entanto, é preciso reconhecer que o inverso também acontece: algumas coisas bem fáceis no Word se revelam um pouco difícil no LaTeX, mas *apenas* no início – e isso é muito importante. Na verdade, creio que o LaTeX tem uma curva de aprendizado bem específica: no início, tudo é bastante difícil; porém, na medida em que nos acostumamos com o funcionamento do LaTeX, tudo fica incrivelmente mais fácil. O primeiro documento irá exigir bastante esforço, mas depois disso quase tudo estará automatizado e não levará mais que alguns segundos para colocar um documento longuíssimo exatamente nas normas desejadas.

Deve-se dizer também que, por ser *open source*, o LaTeX possui uma série de vantagens em relação aos editores comerciais. Além de ser gratuito, há uma vastíssima comunidade online que constantemente amplia seus recursos e auxilia os recém-chegados. A maior vantagem, no entanto, me parece ser sua estabilidade ao longo do anos: não é preciso ficar reaprendendo a utilizar o programa a cada nova versão: depois de aprender os comandos básicos, você irá produzir documentos que irão durar décadas.

Por fim, devo reconhecer o maior defeito do LaTeX: realmente não é fácil *começar* a utilizá-lo. Ele lhe poupará muito tempo e energia no médio prazo, mas exigirá um esforço considerável no início. Para quem possui noções sólidas de computação, esse esforço inicial será pequeno devido

à familiaridade com conhecimentos adjacentes. No entanto, para os pesquisadores que vêm das humanidades, realmente será necessário fazer um esforço inicial maior para adquirir um novo modo de *pensar* sobre seus documentos. Esse livro foi escrito justamente com o objetivo de ajudar nesta etapa: pretendemos levar o leitor absolutamente iniciante a dar os primeiros passos com o LaTeX.

## Visão geral do livro

O objetivo desse livro é ajudar o leitor a *começar* a utilizar o LaTeX. Não pretendo fazer um guia completo sobre LaTeX: há muitos guias extensivos na internet escritos por pessoas com conhecimento técnico mais profundo. No entanto, vejo que meus amigos frequentemente tropeçam justamente nos primeiros passos e acabam desistindo de ir para esses guias mais avançados por lhe faltarem uma série de noções que, muitas vezes, estão apenas implícitas nestes guias. Em outras palavras, o objetivo maior desse guia é servir como empurrão inicial para o leitor interessado no LaTeX.

Para realizar esse objetivo, me ocorreu que seria interessante misturar estilos de exposição e tipos de problemas bem diferentes. Esse livro irá tratar de vários aspectos distintos do LaTeX, alternando entre princípios bastante abstratos e conselhos bastante práticos. Tomei essa decisão porque considero que uma boa dose de teoria é necessário para compreender o LaTeX, mas também é necessário uma quantidade suficiente de prática para manter o usuário interessado. Afinal, por um lado, o LaTeX é um sistema que possui um modo de lidar com os textos bastante diferente dos editores convencionais, o que exige que o leitor desenvolva um novo modo de *pensar* sobre seus textos. Por outro lado, imaginei que teoria demais assustaria o leitor iniciante que – com muita razão – espera poder ir testando como o LaTeX funciona na prática antes de se investir uma quantidade maior de tempo no assunto.

Para realizar esse duplo objetivo, alguns sacrifícios e opções tiveram que ser feitos: não tendo espaço para tratar simultaneamente de todos os princípios teóricos e recomendações práticas, escolhi uma combinação que – em minha opinião – irá atender às necessidades básicas dos pesquisadores acadêmicos. O leitor certamente perdoará uma dose de idiossincrasia: o LaTeX é um sistema poderosíssimo e bastante complexo, com inúmeros de sistemas auxiliares e pacotes de expansão. Entre um universo vastíssimo de possibilidades, escolhi aquelas que, na minha experiência pessoal, se revelaram soluções eficientes para meus problemas. Não nego que existam outras soluções e outras alternativas, mas considero que, para o leitor com objetivos semelhantes aos meus – escrever teses, artigos e preparar apresentações para aulas e congressos –, as soluções exploradas por esse livro serão amplamente suficientes. E, em todo caso, o leitor que quiser caminhar para algo mais avançado, poderá usar esse livro como base para pulos maiores.

Tendo isso em mente, procurei estruturar o livro de modo a fornecer uma visão geral de todas as funções básicas necessárias para atividades acadêmicas. O primeiro capítulo é mais “filosófico”, tentando descrever os principais gerais que diferenciam o LaTeX dos outros editores. No segundo capítulo, alterno para algo extremamente prático: explico como instalar e começar a utilizar o LaTeX. No terceiro capítulo, discuto alguns princípios teóricos sobre o LaTeX, tentando explicar isso que estou chamando de um novo modo de *pensar* sobre os documentos.

Deste ponto em diante, o livro começa a ficar realmente prático. No quarto capítulo, tento explicar o modo o LaTeX efetivamente processa os textos, isto é, como ele transformar os textos

escritos no documento-fonte no documento final que será efetivamente apresentado. O capítulo cinco, em certo sentido, é complementar ao anterior, explicando o modo como os comandos do LaTeX funcionam e afetam o processamento dos textos. Entre esses dois capítulos, o leitor terá os fundamentos básicas para escrever qualquer texto em LaTeX.

Em certo sentido, os capítulos seis e sete são complementares; mas trata-se de um complemento *essencial* para os acadêmicos: o sexto capítulo explica como associar seus documentos em LaTeX com um sistema de administração de referências bibliográficas, enquanto o sétimo explica como criar apresentações. Embora não sejam funcionalidades absolutamente essenciais para o LaTeX, consideramos que serão de interesse especial para o público-alvo deste livro – e, francamente, a economia de esforço derivada dessas duas funções terminam por ser um dos principais argumentos à favor da utilização do LaTeX. Se o leitor chegou até esse ponto, valerá a pena gastar um pouco mais de tempo e resolver essas duas grandes dores-de-cabeça da vida acadêmica: lidar com apresentações e com bibliografias.

Por fim, adicionamos um breve capítulo final, com algumas sugestões bem gerais, e uma seção de anexos, onde o leitor encontrará o código-fonte de todos os arquivos utilizados como exemplo ao longo desse livro.

Dito isto, devo dizer que, embora eu mesmo tenha hesitado antes de adotar o LaTeX – justamente por causa das dificuldades que esse livro pretende resolver –, ele efetivamente se tornou um dos meus principais instrumentos de trabalho. Espero que, ao final deste livro, os leitores possam dizer o mesmo.

# Capítulo 1: Os princípios do LaTeX

## O LaTeX é uma linguagem e não um programa

Reconheço que minha introdução seguiu uma ordem esquisita: comecei apresentando ao leitor o *porquê* de utilizar o LaTeX antes de explicar exatamente *o que* é o LaTeX. O motivo disto é que essa resposta é um pouco mais complicada e precisava de mais tempo para ser desenvolvida. Acontece que, ao contrário do *Word* e do *OpenOffice*, o LaTeX, estritamente falando, *não* é um programa de computador: ele é uma *linguagem de formatação*. Esse é um dos pontos que parecem banais para os alunos de exatas e, por isso, nunca são bem explicados nos guias de introdução ao LaTeX. Por isso, peço aos leitores das humanidades um pouco de paciência, pois vocês logo verão que isso não é tão complicado assim.

Vamos primeiro pensar sobre o que é um programa de computador: em termos bastante simples, um programa é um aplicativo que permite ao usuário fornecer instruções ao computador. Em outras palavras: o programa *realiza* algo. Um editor de texto é um programa que permite ao usuário manipular o texto: você escreve, muda a fonte, a formatação, etc. Depois, o editor *salva* todas essas modificações em um arquivo que pode depois ser aberto e modificado novamente pelo mesmo programa. Ou seja: um editor de texto é um programa que lhe permite manipular um arquivo de texto.

O LaTeX, ao contrário, é uma *linguagem de formatação*: ele é uma série de códigos que indicam as características do documento em que você está trabalhando. Estritamente falando, você pode escrever em LaTeX em qualquer programa – mesmo no *Word* e até mesmo *fora do computador* em uma folha de papel – o que seria bastante esquisito, mas inteiramente possível. Em outras palavras: o LaTeX é um *modo* de lidar com os textos que pode ser realizado por diferentes programas distintos.

Para compreender como isso é possível, precisamos nos acostumar com um dos princípios fundamentais do LaTeX: a separação entre o *conteúdo* do texto a *aparência* do texto. Os editores convencionais misturam essas duas coisas o tempo inteiro. Quando você está trabalhando em um desses editores, você está simultaneamente vendo o conteúdo do texto (as palavras escritas) e a configuração visual do texto (a fonte, o espaçamento entre parágrafos, a margem, etc.). No LaTeX, essas coisas estão *separadas*: ao invés de modificar diretamente a aparência do texto, o LaTeX utiliza uma série de comandos para assinalar como essa aparência deveria ser.

Embora isso pareça muito abstrato, um exemplo mostrará que isso é na verdade bastante evidente. Em um editor de texto, uma palavra itálica aparece do seguinte modo:

A seguinte palavra está em itálico: *itálico*.

Enquanto isso, no LaTeX, o usuário fica diante de duas coisas distantes. De um lado, o leitor possui acesso a uma espécie de *código-fonte* do texto final, isto é, ele lida diretamente com os comandos que determinam a aparência final do documento. Enquanto a *aparência final* do documento será idêntica ao trecho destaco anteriormente, o autor de um documento em LaTeX terá que produzir o seguinte código para produzir o mesmo efeito:

A seguinte palavra está em itálico: \emph{itálico}.

Sei que, nesse momento, o leitor deve estar pensando: porque eu iria preferir que meu texto tivesse esse “\emph” esquisito no meio do texto ao invés de ver imediatamente a aparência final do documento? É aqui que entramos em uma distinção fundamental: no LaTeX, o texto que você escreve é diferente do texto que você lê. O texto final aparecerá em itálico do mesmo modo como aparece no editor de texto, mas o arquivo fonte trará comandos como esse. Essa separação, embora cause alguns inconvenientes *durante* a redação do texto, permite uma série de vantagens para o autor no momento em que for *gerar* o documento final – como veremos, essa separação irá trazer muito mais estabilidade, flexibilidade e controle sobre o texto final.

Por enquanto, o fundamental é compreender o seguinte: rigorosamente falando, o LaTeX não é um programa. Existem diferentes programas que você utilizará para lidar com o LaTeX, mas nenhum deles é absolutamente necessário. O que realmente caracteriza a essência do LaTeX é essa separação entre forma e conteúdo: o LaTeX é uma *linguagem de formatação*, isto é, ele é uma série de convenções que indicam – por meio de comandos como vimos no exemplo acima – qual deveria ser a aparência final ideal do documento final. Em suma, ao trabalhar com o LaTeX, você primeiro precisa criar a estrutura ideal do seu documento a partir de categorias lógicas, para apenas depois *gerar* o documento final.

## Porque separar o conteúdo da formatação

Em minha experiência, quem não tem experiência com programação logo pensa em desistir do LaTeX assim que ouve falar em códigos de formatação. Por isso, imagino que o leitor já esteja começando a duvidar das vantagens em realizar essa mudança. Não seria muito mais simples continuar utilizando um editor tradicional? Porque se dar o trabalho de separar o conteúdo da formatação?

Para responder à primeira pergunta, volto a algo que disse na introdução: a facilidade ou dificuldade do LaTeX depende bastante do objetivo do seu texto. Essa separação entre apresentação e conteúdo realmente dá mais trabalho no início, mas ela traz vantagens enormes para quem precisa produzir documentos mais complexos. Para quem precisa produzir trabalhos acadêmicos, onde é importante seguir um conjunto muito preciso de normas textuais – e onde ocasionalmente é necessário *mudar* as normas seguidas – o LaTeX termina por significar uma enorme economia de esforço, além de garantir documentos de muito maior qualidade.

Para vermos como isso é possível, vamos retornar ao exemplo do trecho em itálico para imaginar um caso em que o código se revela mais vantajoso do que a alternativa. Note-se que o itálico no

LaTex é assinalado pelo comando “\emph” que significa “ênfase”. Digamos que você escreveu um texto bastante longo, utilizou o itálico dezenas de vezes e acaba de ouvir do editor que precisa transformar todas essas passagens para o negrito. No caso de um editor de texto convencional, você precisaria ir olhando todo o documento, palavra por palavra, para encontrar os itálicos e corrigí-los. No LaTeX, você pode redefinir a função de “\emph” com um novo comando e automaticamente mudar toda a aparência do documento.

Note-se também que há um ganho enorme na estabilidade do documento. Em um editor convencional, mesmo se verificarmos palavra por palavra, é possível deixar desapercebido pequenas diferenças de formatação: talvez haja uma mudança não-intencional na fonte de uma passagem, talvez haja um problema no alinhamento de um parágrafo, talvez haja uma mudança no espaçamento entre linhas. Em suma, em um editor convencional, sempre é difícil identificar exatamente o motivo por trás da formatação de cada trecho, pois você precisa olhar passagem por passagem e verificar visualmente se tudo está correto. No LaTeX, como você trabalha diretamente com o código-fonte do documento, então você sempre tem *certeza* dos fatores que estão influenciando a aparência final do seu documento. Não há mistério algum por trás das fontes, do tamanho do texto, das margens, do alinhamento do parágrafo – sempre há um comando específico responsável por cada resultado.

Talvez esses exemplos pareçam forçados Talvez alguns leitores não se importem tanto assim com a aparência final dos seus documentos. No entanto, existe um público que realmente precisa ter esse tipo de controle e que repetidas vezes se encontra diante de situações análogas. Por isso mesmo, o LaTeX se revela tão útil como ferramenta acadêmica: esse é um exemplo – mas não o único – de um público que realmente precisa ter um controle preciso sobre a aparência do documento e que frequentemente precisa realizar mudanças globais na aparência visual de um longo texto.

Por exemplo, digamos que o leitor está terminado uma tese com centenas de páginas e várias centenas de notas de rodapé e inúmeras referências bibliográficas. Dependendo do nível de exigência da banca, ele terá que verificar página por página, nota de rodapé por nota de rodapé, referência por referência para verificar se tudo está com a aparência correta. Além disso, digamos que seu trabalho – escrito de acordo com as normas da ABNT – foi aceito para publicação por uma editora internacional que utiliza as normas da APA. Isso significa que agora o pobre autor precisa *revisar* e *modificar* todas as referências bibliográficas, todas as citações, todas as notas de rodapé, verificando passagem por passagem de todo o documento novamente.

Enquanto isso, o seu colega que produziu sua tese em LaTeX, diante da mesma necessidade, irá fazer todas essas modificações em poucos segundos. Como todas as notas de rodapé, todas citações e todas referências da sua tese estão indicadas por comandos específicos, o autor do texto precisa apenas modificar a *lógica* por trás da aparência do documento para criar imediatamente um documento inteiramente novo dentro das novas normas. O LaTeX utiliza uma série de códigos e comandos para calcular automaticamente o melhor modo de apresentar o texto final de acordo com as normas indicadas pelo próprio autor. Portanto, o colega que tiver usado o LaTeX, não apenas terá a certeza que seu documento estará inteiramente no formato desejado, como essa mudança inteira lhe custará apenas o trabalho de adicionar ao seu documentos algumas breves linhas de código e mandar o LaTeX gerar uma novo documento final.

Essa é apenas uma das várias vantagens causadas por essa separação entre o *conteúdo* e a *apresentação* de um texto. No Word, cada nota de rodapé é relativamente independente: se você

copia e cola um texto sobre ela, a formatação pode ficar ligeiramente diferente das outras. No LaTeX, *todas* as notas de rodapé criadas pelo comando “\footnote{}” irão seguir exatamente a formatação indicada para esse comando – a não ser que você queira deliberadamente alterá-lo por um novo comando. Portanto, você não apenas tem a certeza de que todas elas estão exatamente com a mesma aparência como basta alterar a convenção para esse comando alterar todo o seu texto. Você utiliza um comando para dizer ao LaTeX o que *deveria* acontecer naquela passagem e outro comando para dizer que convenções textuais seguir. Essa separação torna possível mudar toda a aparência do documento com a mera indicação de uma nova convenção. Em suma, não é mais preciso nunca mais ir verificando as margens, o afastamento e a fonte de cada passagem: o LaTeX irá gerar tudo isso automaticamente *a partir* dos critérios que você escolher.

Em outras palavras, os textos produzidos por LaTeX, por definição, sempre seguem automaticamente um padrão de formatação. Enquanto em um editor convencional você precisa se esforçar para seguir um padrão, no LaTeX você precisa se esforçar para *fugir* do padrão. Note-se que essa estabilidade não significa falta de flexibilidade. É exatamente o contrário: no LaTeX sempre existe um comando para mudar a aparência do documento; seja uma mudança pontual, seja uma mudança global. O LaTeX, portanto, é intrinsecamente voltado para produzir documentos padronizados que precisam seguir uma aparência final bastante estrita e, *ao mesmo tempo*, manter a possibilidade de realizar alterações globais nesse padrão sempre que necessário. Por isso, afirmamos que o LaTeX é a ferramenta ideal para quem precisa produzir documentos que sejam simultaneamente *estáveis* e *flexíveis* – o que é exatamente a necessidade do público acadêmico e daqueles que trabalham com documentos institucionais.

Nesse sentido, é preciso reconhecer que um trabalho *inicial* maior no LaTeX: você precisa aprender uma série de comandos e informar ao LaTeX exatamente as características desejadas pelo seu documento. No entanto, esse investimento inicial é amplamente recompensado pelo controle e pela estabilidade do documento final: depois que essas configurações são definidas uma *única* vez, é possível repetir o mesmo modelo indefinidamente e nunca mais pensar na formatação do seu documento.

Note-se ainda que, como o LaTeX é amplamente utilizado no mundo todo, existem uma infinidade de modelos e extensões que podem ser facilmente encontradas na internet. Ou seja: mesmo o trabalho inicial em preparar os parâmetros do seu documento pode ser quase inexistente. Além de ser fácil encontrar modelos para os grandes padrões bibliográficos internacionais, atualmente – graças ao trabalho notável do grupo ABNTex2 e do Lauro César – é possível encontrar até mesmo uma série de pacotes que automaticamente colocam seus documentos inteiramente de acordo com as regras da ABNT.

Por fim, note-se que o LaTeX é flexível o suficiente para você criar suas próprias normas textuais se assim o desejar. Você pode tanto produzir folhas de estilo inteiramente novas ou utilizar os modelos existentes como base e ir progressivamente criando suas próprias modificações. No entanto, certamente uma das maiores vantagens do LaTeX é poder aproveitar o trabalho já produzido por uma imensa rede de colaboradores. Como o LaTeX segue o espírito do *código aberto*, seus inúmeros usuários constantemente ampliam as funcionalidades do programa e dividem suas criações com outros usuários.

Em suma, há realmente um *custo* adicional em separar o conteúdo da aparência de um texto. Cabe ao leitor, no entanto, avaliar se, em seu caso pessoal, esse custo recompensa as seguintes

vantagens trazidas pelo LaTeX: 1. Maior controle sobre a formatação, porque não há regras ocultas do editor bagunçando seu texto; 2. Mais estabilidade ao longo do documento, pois cada recurso estilístico se refere ao mesmo conjunto de comandos; 3. Mais flexibilidade, pois é possível alterar qualquer aspecto do documento; 4. Economia de tempo e esforço, pois é possível importar modelos previamente elaborados que colocam todo o seu documento nas normas desejadas.

Evidentemente, não é em todos os casos que o investimento de tempo exigido pelo LaTeX será recompensado. Para quem escreve raramente, talvez seja preferível utilizar uma ferramenta mais simples. Para quem escreve textos onde a formatação final não é relevante, talvez o LaTeX seja uma ferramenta poderosa demais. No entanto, não consigo imaginar um sistema mais eficiente para quem escreve com frequência e para precisa ter um controle preciso sobre a aparência final do documento. Em suma: para quem precisa escrever relatórios, artigos e preparar apresentações, a complexidade adicional dos códigos será facilmente compensada pelo poder e flexibilidade do LaTeX.

## Como trabalhar com LaTeX

Quando as pessoas começam a utilizar o LaTeX, o primeiro impulso é procurar na internet por editores de LaTeX. Eu considero que isso é um equívoco, pois impede as pessoas de perceberem que o LaTeX *não* é um programa. É verdade que existem programas *para* o LaTeX, mas esses programas *não* são de modo algum necessários para produzir arquivos com o LaTeX<sup>1</sup>.

Como vimos, o LaTeX é uma *linguagem de formatação*; em outras palavras, é um conjunto de códigos convencionais que indicam como deveria ser a aparência de um texto. Qualquer texto que tiver marcações como “\footnote{}” ou “\emph{}” será um texto escrito na linguagem do LaTeX – mesmo se tiver sido escrito em um guardanapo (o que não é nada recomendado!).

Evidentemente, isso cria dois problemas: (i) onde devo escrever esses códigos esquisitos; (ii) como fazer para transformar esses códigos em algo apresentável para alguém que não conhece o LaTeX. Enfim, embora o LaTeX em si mesmo não seja um programa, em algum momento você vai precisar utilizar algum programa para resolver esses dois problemas; ou seja, você precisa de dois tipos diferentes de programas: 1. Um programa que lhe permita *escrever* textos em LaTeX; 2. Um programa que lhe permita *transformar* os arquivos em LaTeX em um formato final apresentável.

Embora eventualmente seja necessário utilizar esses programas, acho importante reforçar a distinção entre esses programas e o LaTeX em si mesmo. Essa distinção é importante porque, em um sentido muito profundo, ela torna todos esses programas opcionais e mesmo descartáveis. Ao contrário de um documento de *Word*, onde seu arquivo fica preso ao programa utilizado (afinal, você precisa não apenas ter o *Word*, mas a versão certa do *Word* para lê-lo adequadamente), seus arquivos em LaTeX podem ser lidos por qualquer programa capaz de ler arquivos em texto puro<sup>2</sup>.

Esse é um conceito meio esquisito para o pessoal de humanas, porque nós aprendemos que cada arquivo possui uma terminação específica que requer um programa específico. Por isso, os arquivos que terminam com “.pdf” devem ser lidos pelo Adobe Acrobat Reader, os arquivos

<sup>1</sup>Eu mesmo não utilizo nenhum editor específico para o LaTeX. Prefiro utilizar o Sublime Text, pois ele é mais flexível e pode servir tanto para LaTeX como para outros formatos.

<sup>2</sup>Na verdade, não existe arquivo em texto puro. Todo artigo é “codificado” de um jeito específico, o que pode ocasionalmente bagunçar os caracteres especiais, como os acentos latinos. Posteriormente, discutirei nisso com mais cuidado. No entanto, fica já uma dica: para evitar problemas com acentos, na dúvida, sempre que vir uma pergunta por “encoding”, sua salvação será “UTF-8”.

“.doc” pelo Word, os arquivos “.pps” pelo Power Point, etc. Isso acontece porque esses arquivos possuem tanto o conteúdo produzido por você como uma série de códigos especiais que só podem ser lidos por esse programa. Por isso, mesmo quando você tenta exportar esses arquivos para outro programa ou mesmo para uma versão diferente do mesmo programa, muitas vezes o arquivo fica bagunçado: perde-se alguns dados e parte da formatação porque *aquele* arquivo foi especificamente codificado para ser lido por *aquele* programa.

O LaTeX, ao contrário, utiliza arquivos de texto puro: ele não tem nenhuma formatação ou codificação especial. Ou melhor: essa codificação existe, evidentemente, mas ela é *produzida por você* e não gerada automaticamente pelo programa. Portanto, o usuário do LaTeX tem acesso direto ao código-fonte do seu arquivo e pode manipulá-lo diretamente, utilizando *qualquer* programa que permita a edição direta do código do arquivo.

Existe a convenção em salvar os arquivos de LaTeX com a terminação “.tex” – o que recomendamos que você faça, para identificar quais os arquivos foram escritos especificamente o LaTeX. No entanto, mesmo se você não seguir essa recomendação, ainda poderá abrir seus arquivos por qualquer editor de textos. Um arquivo de LaTeX possui apenas texto e comandos especiais. Por isso, você deve poder abrir um arquivo de LaTeX em qualquer programa que edite textos puros: até mesmo o bloco de notas daria certo.

Tecnicamente, você pode até mesmo *escrever* em LaTeX no Word ou no Google Docs[3], mas isso não é uma boa opção. Se fizer isso, quando for salvar seu arquivo, o Word (ou equivalente) vai “prender” seu conteúdo em um arquivo “.doc” que só pode ser lido por ele. Portanto, você precisaria utilizar o Word novamente para abrir o arquivo e depois copiá-lo para um programa que possa exportar os arquivos em LaTeX. Portanto, embora seja uma possibilidade teórica, não é algo que faz sentido fazer na prática; é preferível desde o início utilizar programas adequados para essa finalidade.

Resumindo: o LaTeX não é um programa, mas você precisa de dois tipos diferentes de programas para utilizar o LaTeX. O primeiro programa será utilizado para digitar os textos, enquanto o segundo programa será utilizado para transformar o código-fonte do arquivo em LaTeX em algo legível por não-usuários do LaTeX.

## Escrevendo documentos em LaTeX

Sei que insisti demais em repetir que o *LaTeX* não é um editor de texto. No entanto, é realmente inevitável que você utilize *algum* editor de texto para escrever seus arquivos em LaTeX. Nessa passagem, irei explicar que existem três tipos de editores entre as alternativas aceitáveis para trabalhar com LaTeX.

O modo mais simples e rápido de começar a trabalhar com LaTeX é utilizar *um editor de texto puro*. Estou me referindo, com isso, aos editores que *não* introduzem formatação em especial no texto que você digita. Os usuários do Windows podem imediatamente abrir o *Bloco de Notas* e entender o que estou falando. Tecnicamente, você pode começar imediatamente a utilizar esse programa para escrever texto em LaTeX sem qualquer problema.

No entanto, o *Bloco de Notas* não é um bom exemplo de editores de texto simples (nem sempre ele vem configurado para lidar com *unicode*, o que explicarei depois). Se você quiser optar por uma alternativa minimalista, o ideal é utilizar algumas dessas duas alternativas, dependendo do

seu sistema operacional: Notepad++ (para Windows) e TextWrangler (para mac). Esses editores são gratuitos, leves, rápidos e extremamente simples – mas são absolutamente suficiente para produzir seus primeiros documentos em LaTeX.

A segunda alternativa é utilizar um editor de texto *especificamente* voltado para o LaTeX. O TeXworks é um editor que funciona em todos os sistemas operacionais – e é nossa recomendação para o leitor utilizar enquanto ler este livro. O TeXShop é uma alternativa exclusiva para OS X (mac) que acho bastante simpática. Esses editores possuem diferentes graus de complexidade: alguns são apenas versões do *Bloco de Notas* com alguns recursos adicionais específicos para LaTeX, enquanto outros são programas extremamente complexos com inúmeras funções (inclusive, a capacidade de exportar diretamente os arquivos .tex para outros formatos). O TeXworks possui a vantagem de vir automaticamente configurado para salvar seu arquivos em UTF-8 (para ter certeza, vá em “Preferences” e verifique que a seção “Encoding” está selecionada com a opção “UTF-8” – explicarei depois porque isso é importante).

A decisão em utilizar um editor específico para LaTeX é muito pessoal: algumas pessoas gostam de ter a ajuda do programa (que funciona de modo parecido com o Word, com botões para colocar o texto em itálico e uma série de outros atalhos), enquanto algumas pessoas ficam perdidas com as imensas de possibilidades oferecidas por esses programas. Há uma espécie de balança nessa escolha: quanto mais poderoso for o programa, mas ele exigirá de você. Por isso, recomendo que as pessoas testem algumas opções e decidam o que funciona melhor para eles. No meu caso pessoal, sempre me pareceu que esses programas fazem mais coisas do que eu preciso e que tiram o meu foco enquanto estou escrevendo; tenho amigos, no entanto, que só utilizam o LaTeX por causa da ajuda fornecida por esses editores. Mais uma vez: é uma decisão pessoal; cada pessoa terá necessidades e habilidades diferentes e optará por uma opção particular.

Uma grande vantagem em utilizar um *editor de LaTeX* é que ele possui a opção em ativar internamente em ativar o programa que transformar seus arquivos “.tex” em arquivos “.pdf”. No entanto, prestem atenção em um detalhe importante: embora programas como o TeXworks e o TeXShop possam fazer isso, eles precisam que os programas que fazem essa transformação (que discutiremos a seguir) já estejam instalados previamente em seu sistema. Portanto, não adianta baixar um desses programas e achar que possui tudo que precisa para utilizar o LaTeX. Os programas citados anteriormente só podem trabalhar com o LaTeX *depois* que você já tiver uma série de programas auxiliares. No próximo capítulo, veremos com instalar todos os esses programas em conjunto por meio da uma “distribuição” completa do LaTeX.

Uma terceira alternativa é utilizar um editor de programação. Essa opção só faz realmente sentido para programadores ou pessoas com bastante interesse em aprender ferramentas novas. Os programadores geralmente utilizam ferramentas incrivelmente mais flexíveis, mas que também são bem mais difíceis de utilizar no início. A vantagem dessas ferramentas é que elas geralmente possuem uma comunidade *online* bastante ativa que desenvolve extensões para lidar com uma diversidade enorme de tipos de arquivos. Pessoalmente, utilizo o Sublime Text; assim que instalado, ele funciona como um editor de texto extremamente simples, mas ele pode rapidamente se tornar extremamente poderoso com a instalação de alguns pacotes (entre eles, o LaTeXTools, que traz recursos específicos para o LaTeX, como o nome deixa claro). Note-se que existem opções ainda mais difíceis e avançadas – cabe ao leitor escolher o grau de complexidade desejado por ele.

Como já disse: a escolha do editor é bastante pessoal. É importante também lembrar que seus

documentos em LaTeX *não* ficarão atrelados a nenhum editor. Por isso, não há problema algum em utilizar um editor mais simples no início e depois mudar para algo mais avançado. Na verdade, a maioria dos usuários de LaTeX passam por *inúmeros* editores ao longo dos anos. Afinal, uma das vantagens do LaTeX é justamente a flexibilidade de suas ferramentas.

Em todo caso, deixo aqui algumas recomendações para o leitor que se inicia no LaTeX: considero que, para acompanhar os exercícios realizados nesse livro, o TeXworks é melhor opção inicial. Ele é potente o suficiente para fazer tudo que é necessário e simples o suficiente para não ser necessário perder tempo aprendendo a utilizá-lo. Além disso, ele possui a vantagem inegável de funcionar em todos os sistemas operacionais. Depois de terminar este livro, o leitor terá a base necessária para testar sistemas mais avançados e poderá decidir por si próprio qual é o editor ideal. Quando chegar nessa etapa, recomendo que considere o Sublime Text, que pode também lidar com uma variedade vastíssima de tipos de arquivos além do LaTeX.

## Lendo documentos em LaTeX

Como disse antes, a grande vantagem do LaTeX é a segurança de que a aparência final do documento está inteiramente sob seu controle. Entretanto, os documentos escritos em LaTeX estão repleto de códigos estranhos. Cabe, portanto, a pergunta: como o arquivo escrito em LaTeX se torna legível para o público em geral?

Para isso, o LaTeX (a linguagem de formatação) utiliza o TeX (um sistema de tipografia). A semelhança entre os nomes não é coincidência: o TeX é o irmão mais velho, um sistema de tipografia inventado por Donald Knuth na década de 70. O LaTeX é uma linguagem de formatação mais recente – da década de 80! – que adiciona novos recursos ao TeX. Não é necessário compreender exatamente como o TeX faz essa transformação. Basta saber que ele consegue entender seu arquivo e transformá-lo em uma documento final segundo a aparência e o formato desejado.

Na verdade, o TeX não trabalha mais sozinho. Ao longo dos últimos trinta anos, vários sistemas auxiliares foram desenvolvidos para ampliar ainda mais suas funcionalidades. Mas não é necessário se aprofundar nesses detalhes. O fundamental é saber que, em cima desse sistemas consideravelmente antigos, há uma série de programas que conseguem transformar seu documento inicial em basicamente qualquer formato que lhe interesse. Todos esses programas responsáveis em *gerar* seu arquivo final são gratuitos, possuem o código aberto e geralmente vêm todos juntos quando você instala uma “distribuição” completa do LaTeX.

Não é necessário conhecer todos os detalhes sobre como esse processo ocorre. Basta saber que tanto o TeX como suas variantes podem ser baixadas e instaladas facilmente em um pacote completo. Aliás, há mais um nome que vale a pena lembrar: o XeTeX. Ele é um sistema de formatação adicional que torna mais fácil a utilização de caracteres internacionais (como nossos acentos latinos e coisas mais exóticas como o grego polifônico). Se entre o arquivo inicial e a apresentação final do seu documentos todos os acentos forem bagunçados, a solução provavelmente será utilizar o XeLaTeX (a combinação entre XeTeX e LaTeX) para gerar seus arquivos finais.

Caso tudo isso tenha parecido muito estranho, peço que o leitor não se assuste: na prática é incrivelmente simples. Tudo que você precisa saber é o seguinte: 1. Para usar o LaTeX, você deve

escrever um arquivo em texto puro, com os códigos de formatação e salvá-lo com a terminação “.tex”. 2. Quando seu arquivo estiver pronto, você irá ativar um programa que vai “compilar” esse arquivo e transformá-lo em um documento final legível por qualquer pessoal. Geralmente, a opção atual mais comum é gerar um arquivo em formato PDF”.

Esses programas que fazem a “tradução” entre um “arquivo.tex” e um “arquivo.pdf” normalmente podem ser ativados diretamente pelo seu editor em LaTeX. No entanto – sei que me repito, mas isso é importante –, não é próprio editor que faz essa conversão. Seu editor de texto irá apenas “chamar” um programa externo que faz essa conversão. É importante repetir isso para o leitor compreender que não é suficiente instalar um editor de LaTeX para utilizar o LaTeX. O editor de LaTeX precisa que esses “conversores” já estejam instalados para funcionarem normalmente – e, mais vezes, veremos como instalá-los no próximo capítulo.

Embora talvez isso pareça uma complicação um pouco assustadora, quero tranquilizar o leitor e dizer que *realmente* não é necessário compreender como esses compiladores funcionam. É preciso apenas instalá-los que eles farão esse trabalho automaticamente por meio do seu editor de LaTeX.

No entanto, o que é necessário é entender que vários sistemas auxiliares podem ser utilizados. Se você já estiver com o TeXworks ou TeXShop instalado, dê uma olhada na barra de atalhos: lá você irá encontrar um menu com várias opções de compiladores (pdfTeX, XeTeX, LuaTeX, etc.). Todos esses nomes estranhos – e curiosamente parecidos – são versões modernas desenvolvidas a partir do TeX original, pelos próprios usuários que queriam funções adicionais. A *única* coisa que você precisa saber sobre esses compiladores é o seguinte: você precisa escolher um deles para gerar seus documentos.

Existem várias alternativas possíveis, mas não acho necessário o leitor se preocupar com isso neste momento. Em minha experiência, o “XeTeX” tem se mostrado suficiente para resolver todos os meus problemas e, por isso, o utilizei para gerar todos os modelos ao longo deste livro. Portanto, recomendo que o leitor não se preocupe muito com isso e que, ao menos enquanto estiver lendo este livro, simplesmente vá nesse menu e escolha a opção “XeTeX”.

Como explicaremos no próximo capítulo, o XeTeX geralmente lida melhor com acentos latinos e com os caracteres exóticos que os pesquisadores de humanas ocasionalmente precisam utilizar. Depois que terminar o livro, o leitor terá conhecimento suficiente para pesquisar outros sistemas e verificar se não existe alguma alternativa superior para suas necessidades pessoais. Por enquanto, o XeTeX será mais do que suficiente.

## Notas finais

Por fim, vamos revisar os pontos principais discutidos neste capítulo:

1. O LaTeX é uma *linguagem de formatação* e não um programa. Ou seja: praticamente qualquer editor pode ser utilizado para escrever artigos em LaTeX, desde que tragam uma combinação entre texto puro e códigos de formatação.
2. Essa linguagem de formatação serve para separar o *conteúdo* do texto da *aparência* do texto. Essa separação oferece mais controle sobre nosso documento final.

3. Existem diferentes programas necessários para trabalhar com LaTeX. O leitor irá precisar de dois tipos de programas bem diferentes: (i) um programa para *transformar* os arquivos de LaTeX (“.tex”) em arquivos legíveis por qualquer pessoal (“.pdf”); (ii) um programa *escrever* os arquivos em LaTeX.
4. Existem inúmeras opções de editores de LaTeX e cada uma possui suas vantagens. No entanto, para praticar os exercícios deste livro, recomendamos que o leitor comece o TeXworks.
5. Para *transformar* os arquivos de LaTeX, o leitor também possui várias opções. No entanto, e novamente para simplificar, recomendamos que o leitor utilize o XeLaTeX. Não é preciso aprender *como* o XeLaTeX funciona: basta lembrar de selecioná-lo no menu do TeXworks quando for “compilar” seus documentos.
6. O XeLaTeX pode ser *ativado* pelo TeXworks, mas não pode ser instalado pelo TeXworks. Para instalar o XeLaTeX (e todos os outros sistemas auxiliares relacionados ao TeX), o melhor jeito é instalar uma “distribuição” completa do LaTeX. Veremos como fazer isso no próximo capítulo.

# Capítulo 2: Os primeiros passos

## Instalando o LaTeX

Antes de mais nada, devo lembrar que esse livro não pretende *substituir* os manuais de LaTeX, mas apenas esclarecer as noções básicas que estão implícitas nele e que talvez não sejam facilmente acessível para um leitor de formação humanística. Repito isso nesse momento para lembrar ao leitor que é preciso distinguir entre a aprendizagem de princípios fundamentais que permanecem basicamente inalterados ao longo do tempo e procedimentos transitórios que mudam a cada nova versão do LaTeX e de cada sistema operacional.

O modo de instalação do LaTeX entra na segunda categoria: é o tipo de coisa que irá sempre mudar com o tempo e, por isso, é um tema mais apropriado para um guia *online* que seja constantemente atualizado do que para um livro introdutório que pode passar bastante tempo sem atualizações. Portanto, minha primeira recomendação sobre *como instalar* o LaTeX é sempre buscar as informações mais recentes disponíveis na internet. Em todo caso, antes de fazer isso, acredito que o leitor precisa ter algumas informações básicas que explicarei a seguir.

No capítulo anterior, expliquei que o LaTeX é uma linguagem de formatação que utiliza uma série de sistemas de tipografia auxiliares (TeX, XeTeX, etc.); afirmei também que seria necessário possuir um editor de textos e que havia uma série de opções válidas. Espero que a perspectiva de precisar de todas essas coisas diferentes não tenha assustado ninguém. Por incrível que pareça, é relativamente simples conseguir instalar tudo isso simultaneamente em seu computador.

Isso é possível devido ao que se chama de *distribuição*. Esse termo é utilizado para designar um pacote de softwares da mesma família (com todos os requisitos necessários) prontos para ser instalados. Esses arquivos costumam ser significativamente pesados, mas possuem praticamente tudo que você precisa para começar imediatamente a trabalhar com LaTeX: de uma só vez, você instala o TeX, o XeTeX e uma série de programas que talvez se revelem úteis (como programas de administração de bibliografia, por exemplo), além de vários editores de LaTeX. Essas *distribuições* estão disponíveis gratuitamente na internet e são atualizadas periodicamente.

Embora essas distribuições normalmente tragam programas que você talvez nunca chegue a utilizar, *ainda assim* é preferível instalar o LaTeX por meio delas. Lembre-se que o LaTeX originalmente era capaz de rodar em um computador da década de 80! Por isso, esses programas auxiliares tendem a ser bastante leves, de modo que é preferível instalar uma distribuição completa e ter várias opções do que utilizar.

Na página central do LaTeX (<http://www.latex-project.org/>) você encontrará links para cada uma dessas distribuições. Você deve escolher a versão adequada para seu sistema operacional. Não irei explicar os procedimentos de instalação, porque provavelmente vão mudar a cada nova versão. No entanto, é relativamente simples: é só ter paciência para baixar o arquivo, seguir os procedimentos de instalação e aguardar enquanto ele instala os vários programas que vem no pacote.

## Escolhendo um editor

Terminada a instalação, você perceberá que possui agora uma grande variedade de programas relacionados ao LaTeX. O leitor pode se sentir à vontade para testar os editores que vieram com sua instalação. Em todo caso, recomendo que verifique se o TeXworks veio com sua distribuição e, se não tiver vindo, que o instale separadamente. O TeXworks possui duas vantagens para usuários iniciantes: é bastante leve e é simples ativar as duas configurações iniciais que precisamos: que os arquivos sejam gravados em formato de codificação “UTF-8” e que o XeLaTeX seja utilizado para compilar os arquivos.

Para verificar se as duas opções estão corretamente selecionadas no TeXworks, faça o seguinte: (i) primeiramente, vá na seção de “preferências” (Preferences) e procure a opção “Encoding” e verifique que está em “UTF-8”; (ii) em segundo lugar, vá na barra de tarefa no topo do programa e procure um menu com uma lista de sistemas tipográficos (normalmente a opção pdfLaTeX aparece selecionada) e escolha a opção “XeLaTeX”.

Caso o leitor queira utilizar outros editores não há nenhum problema, *desde* que se lembre dessas duas opções de configuração: codificação (encoding) como UTF-8 e XeLaTeX como sistema tipográfico de compilação.

Em todo caso, recomendo que – ao menos no início – o leitor utilize um editor que possa também transformar seus arquivos em LaTeX em arquivos “.pdf”. Como vimos no capítulo anterior, nenhum editor pode fazer isso sozinho, pois ele precisa de sistemas tipográficos (como o XeLaTeX) previamente instalados. No entanto, se você tiver seguido a recomendação acima e instalado uma distribuição completa do LaTeX, já terá todos esses sistemas instalados e seu editor poderá fazer isso sem qualquer problema.

Não é absolutamente necessário fazer essa transformação dentro do seu editor. No entanto, principalmente no início, acho que é uma opção interessante: significa que o usuário não precisa se dar o trabalho em descobrir como ativar o XeLaTeX diretamente. Novamente, recomendamos o TeXworks para isso por ser bastante simples: basta apertar o pequeno botão de “play” na barra de atalho para gerar o arquivo final em formato PDF.

Evidentemente, na medida em que se aprofundar mais no estudo do LaTeX, o leitor poderá testar soluções alternativas àquelas sugeridas neste livro. Por enquanto, no entanto, recomendamos que siga os seguintes passos: 1. Procure uma distribuição completa do LaTeX e instale todas as suas dependências; 2. Instale o TeXworks e utilize esse editor para escrever seus arquivos em LaTeX; 3. Vá na seção de preferências do TeXworks e veja se a opção “Encoding” está marcada em UTF-8; 4. Vá no menu da barra de atalho e troque a opção atual (provavelmente será “pdfLaTeX”) pela opção “XeLaTeX” 5. Quando quiser transformar seus arquivos “.tex” em arquivos “.pdf”, vá na mesma barra de atalho e aperte o botão verde.

## O problema dos caracteres especiais (ou: só unicode salva)

Preciso confessar, caro leitor, que simplifiquei demais algo importante no capítulo anterior: disse que o LaTeX utiliza textos puros. Em minha defesa, isso é uma mentira em que todos nós que

saímos de uma faculdade de humanas acreditamos. A verdade, entretanto, é que não existe texto puro em computação.

Aliás, isso não é de modo algum um problema do LaTeX: é algo muito mais antigo e universal no modo como lidamos com computadores. O leitor certamente já teve a experiência de abrir um arquivo, receber um email ou preencher um formulário na internet e subitamente ver todos seus acentos transformados em caracteres estranhos. Esses pequenos fenômenos mostram que – mesmo hoje – a relação entre os textos e os computadores possuem algumas complicações sérias.

Esse tipo de problema também *pode* ocorrer com o LaTeX, mas há dois tipos de soluções: uma permanente e facilíma e outra provisória e complicadíssima. E aqui repito mais um problema que encontro com os guias online de LaTeX: como a maioria dos usuários padrões de LaTeX são matemáticos (que fazem artigos com muitos números e pouco texto), esses guias geralmente não explicam direto a solução para os acentos. Alguns ainda fazem o desserviço de propor a solução complicadíssima (e provisória) de colocar um código especial para transcrever corretamente cada acento. Evidentemente, esse tipo de solução só é custosa demais: se você precisa escrever algumas dezenas de páginas, é completamente absurdo reaprender a ortografia de todas as palavras, substituindo todos os acentos por códigos especiais.

Felizmente, há uma solução bastante simples e permanente: compilar seus artigos com o XeLaTeX. O XeLaTeX é um desses “programas auxiliares” que eu citei que é automaticamente instalado em uma distribuição do LaTeX. Ele foi desenvolvido justamente para dar conta de vários caracteres internacionais como os nossos acentos latinos. Ele possui ainda uma vantagem importantíssima para os pesquisadores de humanidades: ele também consegue lidar com os caracteres mais exóticos possíveis, passando do grego antigo ao hebraico. Em suma, ele é a solução para os nossos problemas e é muito mais fácil de utilizar do que as soluções padrões mais comumente encontrada em guias para iniciantes.

Isso é tão importante que que peço que o leitor tenha a paciência em dedicar mais algumas parágrafos à compreender exatamente por o XeLaTeX é necessário e porque ele funciona. Esse problema dos acentos é tão irritante que leva muitas pessoas a desistirem do LaTeX ao verem seus primeiros textos saírem truncados. E, apesar desse problema reaparecer em diferentes lugares da internet – formulários, emails – ele é incrivelmente pouco compreendido. Eu mesmo perdi bastante tempo navegando por falsas soluções até compreender o problema – e espero agora poder poupar ao leitor bastante energia e frustração no futuro.

A raiz do problema está na mentira citada anteriormente: a de que existem textos puros. Na verdade, toda vez que o computador lida com textos, ele transforma aquilo a linguagem que nós compreendemos em uma outra linguagem que lhe permite manipular os dados. O problema surge justamente no momento em que é feita essa “tradução” entre a *nossa* linguagem e a linguagem do computador.

Para que essa tradução ocorra, os computadores utilizam um sistema específico para armazenar os dados. Esse sistema é chamado de “codificação” (encoding) do texto. E aqui é que começa o problema: não existe uma codificação universal. Ao longo da história das ciências da computação, diferentes sistemas de codificação foram propostos. No início, isso não era tão problemático, mas na medida em que esses sistemas começaram a se comunicar com mais frequência (ou seja: com a internet), os problemas também começaram a aumentar.

Os primeiros sistemas de codificação foram desenvolvidos tendo-se em mente basicamente

as necessidades da língua dos próprios criadores de cada sistema. Por isso, os americanos desenvolveram um sistema de codificação (ASCII) que se tornou extremamente utilizado – dado à influência do país no desenvolvimento de novas tecnologias – mas que não sabe interpretar acentos latinos, pois estes não eram utilizados em sua língua. Isso levou aos outros países a criarem sistemas paralelos para dar conta dos *seus* caracteres, repetindo e multiplicando o problema: cada novo sistema de codificação dava conta apenas dos seus caracteres, mas não sabia se comunicar com os demais (a codificação latina não compreendia o alfabeto grego que não compreendia o russo, etc.).

Mais uma vez: não estamos falando ainda do LaTeX. Esse era um problema geral da codificação dos textos em todos os computadores que ficou mais evidente quando os emails e os sites começaram a atravessar os continentes, deixando os problemas de compatibilidade se tornarem ainda mais evidente. Os navegadores da internet tentam adivinhar o sistema de codificação original automaticamente se adaptam às suas próprias regras. Mesmo assim, no entanto, erros podem acontecer: se você começou a preencher um formulário e depois seus acentos sumiram ou se tornaram incompreensíveis, é bastante provável que o sistema de codificação do formulário foi desenvolvido para um língua que não possuía esses acentos.

Mesmo em relação aos editores convencionais de grande sucesso comercial, esse problema ainda não foi inteiramente superado. Quem trabalha na área de filosofia antiga sabe que inserir o alfabeto grego pode causar dores de cabeça consideráveis: tudo está ótimo até o arquivo ser repassado para um amigo que não consegue visualizar nada do texto original. Por isso, surgiu o hábito, entre os pesquisadores que lidam com línguas antigas, a usarem *transliterações* das citações originais, isto é, optarem uma convenção para verter o alfabeto estrangeiro para o nosso alfabeto.

Mas não há motivo para desânimo: como disse acima, há uma solução simples e permanente para esse problema. Surgiram sistemas modernos de codificação que permitem utilizar, no mesmo texto, praticamente todos os mais diferentes caracteres possíveis utilizados pelas línguas humanas. Esses novos sistemas de codificação se chama “unicode” e um deles em particular – o UTF-8 – é inteiramente compatível com o LaTeX e com praticamente todas as necessidades “alfabéticas” que o leitor venha a ter.

Enfim, depois dessa longa explicação, chegamos a uma boa notícia: o LaTeX não apenas aceita os acentos latinos diretamente, como também aceita uma quantidade vastíssima de caracteres estrangeiros que provavelmente até mesmo seu editor atual possui dificuldades em utilizar. Na verdade, é *mais fácil* utilizar o LaTeX para lidar com caracteres estranhos do que praticamente qualquer outro sistema. Isso acontece porque o LaTeX possui a capacidade de lidar diretamente com arquivos em UTF-8, o que garante que seu texto final sairá exatamente do jeito desejado<sup>3</sup>. A única coisa que é preciso fazer é avisar ao LaTeX que você irá utilizar esse sistema – e veremos como fazer isso na seção seguinte.

---

<sup>3</sup>Em uma nota pessoal, acho bem mais agradável poder utilizar o alfabeto grego polifônico em meus trabalhos do que ficar lidando com diferentes regras de transliteração. Entretanto, antes do LaTeX, isso era tão trabalhoso que eu nem considerava seriamente a possibilidade de fazer isto sistematicamente. Depois do LaTeX, no entanto, isso ficou tão fácil que não consigo nem imaginar voltar para um editor convencional.

## A solução permanente para os acentos e caracteres estrangeiros

Espero que os leitores que tenham ficado entediados com a explicação mais técnica anterior ao menos parem aqui para ver, na prática, como resolver permanentemente o problema dos caracteres especiais.

A solução tem duas partes: primeiro, você tem que garantir que seu arquivo vai realmente *salvar* esses caracteres; em seguida, você tem que utilizar um sistema que saiba *preservar* esses caracteres na versão final do seu documento. Ambas as partes são incrivelmente simples – especialmente comparado com o fato de que esse é um problema tão antigo e tão chato.

Para garantir que seu documento irá salvar todos seus caracteres corretamente, você precisa verificar, na seção de “preferências” do seu editor, se ele está salvando os arquivos na codificação (encoding) UTF-8. Essa codificação é um tipo de unicode amplamente aceito e que irá reconhecer praticamente qualquer caractere que você utilize em seu trabalho.

(Detalhe relevante: se você seguir minha recomendação e utilizar o TeXworks ou o Sublime Text, você verá que o UTF-8 já é a opção padrão. Portanto, você não precisa fazer nada para utilizar corretamente esses editores.)

Note-se que, estritamente falando, esse primeiro passo ainda não tem a ver com o LaTeX. Se seu editor não salvar os arquivos no formato ideal, você não dará uma chance ao LaTeX em interpretar o texto corretamente, pois o próprio editor irá bagunçar os caracteres na hora de salvá-lo se ele não estiver configurado para utilizar a codificação adequada. Ou seja, embora na sua tela inicial você veja tudo corretamente, essas informações serão perdidas quando o arquivo for salvo. Por isso, *antes* mesmo de começar a escrever, verifique se seu editor está configurado para lidar com UTF-8.

A segunda parte da solução é garantir que, no processo de conversão do arquivo “.tex” original para a sua forma de apresentação (geralmente em “.pdf”), todos os acentos sejam mantidos. É aqui que entra o XeLaTeX. Ele é um programa que complementa o TeX original com as funcionalidades do LaTeX e a capacidade do XeTeX em interpretar textos codificados em unicode (reconheço que esses nomes extremamente parecidos atrapalham mais do que ajudam, mas não é preciso decorá-los).

O importante é saber que o XeLaTeX é o “motor” que transforma seus arquivos “.tex” e que ele possui a capacidade de interpretar corretamente arquivos gravados em “UTF-8”. Em qualquer editor de LaTeX, você terá várias opções de sistemas (pdfLaTeX, LuaTeX, etc.); ignore todos esses nomes *por enquanto* e se lembre de escolher XeLaTeX.

Enfim, por mais complicado que isso pareça, a solução pode ser resumida em dois passos bastante simples: 1. Salve seus arquivos no formato UTF-8; 2. Utilize o XeLaTeX para processá-los.

Se até isso parecer complicado (se você não encontrar onde modificar essa opção), existe uma outra alternativa: você pode adicionar essas duas linhas de códigos ao *íncio* do seu documento e elas tentarão “forçar” seu editor a interpretar seu documento do jeito desejado (note-se que não é sempre que isso funciona).

```
% !TEX TS-program = xelatex  
% !TEX encoding = UTF-8 Unicode
```

Por fim, note-se que a opção pelo XeLaTeX possui algumas consequências. Como veremos, o LaTeX possui uma série de comandos e “pacotes” que são específicos para cada compilador utilizado. Os exemplos desse livro pressupõe que o leitor está utilizando o XeLaTeX.

Por isso recomendamos que, ao menos quando estiver trabalhando nas lições discutidas aqui, o leitor utilize o TeXworks. Ele tornará fácil salvar seus arquivos em “UTF-8” e selecionar o “XeLaTeX”. Depois que terminar o livro, o leitor terá conhecimento suficiente para, se quiser, testar outros sistemas e editores e irá encontrar pacotes e classes compatíveis para cada necessidade.

## Notas finais

Vamos revisar os pontos principais discutidos nesse capítulo

1. Lembre-se que as instruções de instalação podem mudar a cada versão, por isso procure informações atualizadas antes de proceder com a instalação.
2. Lembre de instalar uma *distribuição completa* do LaTeX, isto é, um pacote que virá com um instalador que trará todos os sistemas relacionados com o LaTeX, assim como uma série de programas auxiliares.
3. Procure a distribuição apropriada para seu sistema operacional no site (<http://www.latex-project.org/>)
4. Escolha um editor que possa utilizar automaticamente o sistema de compilação do LaTeX (recomendamos o TeXworks).
5. Lembre-se de salvar seus arquivos no formato de codificação (encoding) UTF-8.
6. Lembre-se de utilizar o XeLaTeX para compilar seus documentos
7. Se for utilizar outra codificação e outro sistema de compilação, verifique a compatibilidade entre pacotes e comandos.

Como o objetivo deste livro é principalmente apresentar os princípios do LaTeX, não irei especificar exercícios que devem ser feito pelos leitores. No entanto, creio que a leitura do resto do livro será mais produtiva se o leitor já estiver com uma instalação do LaTeX em seu computador e puder ir fazendo experimentos para ver como o LaTeX funciona na prática.

# Capítulo 3: Os elementos básicos de um documento em LaTeX

## Introdução

Esse capítulo irá lhe guiar na produção de um documento básico. Precisamos lembrar algo que foi dito na introdução: o LaTeX requer um considerável investimento inicial, mas ele se torna progressivamente fácil quanto mais você o utiliza. Nesse sentido, o documento mais difícil será justamente o primeiro. Você terá que adquirir um novo modo de *pensar* sobre seu texto e terá que aprender uma série de comandos novos. No entanto, depois que você criar esse primeiro documento, ele poderá ser reutilizá-lo inúmeras vezes. Se o leitor tiver a paciência de vencer os primeiros obstáculos, os próximos serão consideravelmente mais fáceis.

Vamos primeiramente relembrar os princípios absolutamente essenciais compreender o que você irá fazer ao elaborar um documento em LaTeX.

1. O LaTeX não é um programa, mas uma *linguagem de formatação*. Isso significa que você pode utilizar qualquer editor de texto, desde que adicione os códigos específicos do LaTeX ao documento.
2. Quando você tiver um documento pronto, escrito com a linguagem do LaTeX, você precisará de um programa para convertê-lo em um documento que seja legível por qualquer pessoa. Em geral, o formato de destino preferido é do Adobe Acrobat Reader, por poder ser lido em qualquer computador.
3. Para realizar essa transformação, não é suficiente ter um editor de texto instalado, você também precisa instalar o sistema de tipografia TeX e seus programas auxiliares. O modo mais simples de fazer isso é instalar uma *distribuição* do LaTeX que irá instalar tudo que você precisa em um único pacote.
4. Depois que estiver com a distribuição em seu computador, você deve escolher um editor de textos para escrever seus artigos. Recomendamos o TeXworks para o início, pois ele é um *editor integrado*: ele mesmo “chama” o XeLaTeX para fazer a transformação do seu arquivo “.tex”. Portanto, ele é o único programa que você irá precisar aprender a utilizar no momento.
5. É importante que você salve seus artigos em UTF-8 e utilize o XeLaTeX para gerar seus arquivos. O TeXworks salva todo arquivo em UTF-8 automaticamente; mas, se você optar por outro editor, terá que verificar isso pessoalmente. O TeXworks também permite escolher o XeLaTeX (e várias outras opções) facilmente, ao disponibilizar um menu na barra de atalhos. No entanto, é possível “forçar” a utilização do XeLaTeX colocando essas duas linhas no início de qualquer documento em LaTeX:

```
% !TEX TS-program = xelatex
```

```
% !TEX encoding = UTF-8 Unicode
```

O restante desse capítulo irá presumir que você já instalou uma distribuição do LaTeX e que está utilizando um editor integrado como o TeXworks.

## Pensando como o LaTeX

O maior obstáculo no aprendizado do LaTeX é internalizar um modo diferente de pensar sobre os textos. Como afirmamos anteriormente, a maior diferença entre o LaTeX e um editor convencional é a *separação* entre forma e conteúdo de um texto. Se você abrir o Word, por exemplo, você já irá ver seu texto do modo como o leitor irá acessá-lo. No LaTeX, você trabalha em cima de um texto entrelaçado por uma série de códigos de formatação que depois irão gerar o texto final de acordo com esses códigos.

A vantagem dessa separação é dupla: (i) você tem maior controle sobre a aparência final do texto (pois pode alterar diretamente os códigos) e (ii) ao escrever, você pode esquecer inteiramente a questão da aparência e se concentrar apenas no conteúdo (pois você sabe que o LaTeX irá resolver tudo isso depois).

Inicialmente, as pessoas acham desconfortável ter que lidar com esses códigos diretamente. No entanto, esse desconforto costuma passar rapidamente. Se não passar, há editores que lhe permitem trabalhar com duas janelas: de um lado, você vê os códigos; do outro, você vê o texto em sua forma final. Há também editores que criam combinações de cores diferentes para ficar mais fácil identificar onde o texto termina e os códigos começam. Nesse sentido, aliás, vale a pena testar diferentes editores, tentando descobrir o que funciona melhor no seu caso.

Em todo caso, o aspecto mais importante disto é que você precisa aprender a separar mentalmente o conteúdo do aspecto visual do texto. Para isso, sua relação com o texto irá mudar: você não precisa ficar ajeitando o parágrafo, ficar pensando nas quebras de páginas, mudando a fonte, etc. Você precisa pensar apenas na relação lógica entre os elementos internos do texto: Preciso criar um novo parágrafo para essa passagem? Preciso inserir uma referência bibliográfica nesse ponto? Devo enfatizar esse termo? Devo inserir uma citação?

Seu texto será um composto de texto puro e uma série de códigos que assinalam a *lógica* por trás de cada seção. Essas marcações lógicas serão depois processadas pelo LaTeX de acordo com as configurações que você determinou para o seu documento. Portanto, ao invés de se preocupar se o título de uma seção está grande ou pequeno, você deve esquecer disto no momento: apenas assinale que ali começa uma nova seção ou uma subseção com determinado título e a formatação será resolvida depois.

Em suma, ao escrever um texto em LaTeX, você deve tratar separadamente a *estrutura lógica interna ao texto* e sua *aparência final*. Por “estrutura lógica” me refiro aos elementos que estarão presente em seu texto independentemente das normas de formatação (título do documento, autor, citações, referências, seções, índices) e por “aparência” falo das margens, das fontes, da formatação do texto, etc. Ao lidar separadamente com essas coisas, o LaTeX permite um controle maior sobre todo o documento.

# Os elementos básicos que compõe um documento em LaTeX

Já explicamos que um princípio central do LaTeX é separação entre o *conteúdo* e a *aparência* do texto. Vamos agora ver como isso funciona na prática. Para realizar essa separação, há duas diferenças principais entre um documento em LaTeX e um documento convencional: (i) o documento em LaTeX possui uma série de *códigos* misturados com o texto; (ii) o documento em LaTeX deve seguir uma *estrutura* bastante específica. Explicaremos esses dois princípios nas seções seguintes.

## Introdução aos comandos (`\comando{variável}`)

O leitor certamente já sabe que o LaTeX requer o uso de comandos especiais para designar a aparência e a estrutura lógica do texto. Na introdução deste livro, utilizamos um exemplo que vamos retomar agora:

No documento *inicial* teremos a seguinte frase: A> Essa frase possui uma palavra em `\emph{itálico}`.

No documento *final* teremos a seguinte frase: A> Essa frase possui uma palavra em *itálico*.

No primeiro exemplo, temos o arquivo em LaTeX tal como ele aparece para nós: ao invés de *vermos* a palavra em itálico, vemos apenas o código que indica que aquela palavra deve ser enfatizada. No segundo exemplo, temos a frase que aparecerá no documento final: o LaTeX interpreta aquele comando e faz com que a aparência da palavra apareça agora em itálico.

Esse exemplo nos apresentou a primeiro comando de formatação do LaTeX: o comando “`\emph{}`”. Esse comando deixa todo o texto que aparecer dentro das chaves (“`{}`”) enfatizado – o comportamento padrão da ênfase é deixar o texto em itálico, mas isso pode variar com o contexto e com outros comandos do usuário.

Extrapolando a partir desse exemplo, podemos agora começar a perceber a estrutura básica de todos os principais comandos do LaTeX: ele começa com a barra invertida (“`\`”) seguida pelo nome do comando (“`emph`”, nesse caso, que designa a noção de *ênfase*), podendo *opcionalmente* ser complementado com valores adicionais definido entre colchetes (“`[]`”) ou entre chaves (“`{}`”). Portanto, um comando típico do LaTeX segue o seguinte modelo:

```
\comando[variável secundária]{variável principal}
```

Diante disso, a maior preocupação dos usuários iniciantes é com a necessidade de *decorar* um comando para cada característica visual do documento e ainda ter que digitá-los novamente a cada vez que for utilizá-los. Embora essa tarefa, à primeira vista, pareça exageradamente trabalhosa, há vários fatores que a tornam consideravelmente mais leve. Em primeiro lugar, logo descobrimos que não há muitos comandos que utilizamos repetidamente (em um artigo acadêmico, utilizamos basicamente os comandos para itálico, nota de rodapé, citação e referência

bibliográfica); em segundo lugar, o editor utilizado pode tornar essa tarefa totalmente automática (meu editor possui um atalho de teclado para as quatro funções que uso repetidamente); por fim, não é realmente necessário *decorar* os comandos incomuns, pois é sempre muito fácil encontrá-los na internet ou nos guias de LaTeX.

Outro ponto importante: noto que é muito comum para os colegas de humanas recuarem diante desses comandos por julgar que seu texto ficará muito estranho ao se ver permeado por esses códigos. Evidentemente, há um aspecto estético na literatura que é muito importante para algumas pessoas. No entanto, é nesse ponto que a utilização de um editor adequado possui uma enorme influência: em alguns editores, é possível alternar entre os códigos e a aparência final quase instantaneamente; em outros, os códigos possuem cores especiais que os tornam distintos do texto. Novamente, no entanto, entramos em uma questão pessoal: cada um terá que decidir o quanto ver diretamente os códigos lhe incomoda e encontrar seu modo preferido de lidar com eles.

Por enquanto, não iremos aprender novos comandos. O fundamental é entender sua estrutura básica: *barra invertida, nome do comando, variáveis entre chaves ou colchete*. No caso do exemplo utilizado, a variável utilizada é o termo que queremos que seja enfatizado. Note-se que não é exatamente um comando para itálico, mas para ênfase: podemos depois redefinir esse comando para que a palavra fique sublinhada ou em negrito.

Ainda para o público acadêmico, esse comando traz vantagens adicionais. Frequentemente precisamos utilizar uma citação que, no original, trazia um termo em itálico. É sempre bastante delicado mover essa citação entre arquivos, pois quando selecionamos, justificamos e limpamos nosso texto, as características da citação original terminam por ser redefinidas. Como os acadêmicos bem o sabem, é sempre necessário retornar para cada citação e verificar se nossa formatação não afetou o texto citado. A utilização do comando “\emph” resolve isso permanente: não importa o que aconteça com o resto do texto, aquele trecho continuará enfatizado.

Note-se que o LaTeX é inteligente o suficiente para adaptar a ênfase ao contexto do parágrafo. Se você colocou toda uma passagem em itálico, a ênfase de um trecho específico fará automaticamente com que apenas a passagem enfatizada saia do itálico. Em suma, você pode utilizar o comando “\emph” dentro de outro comando “\emph” no mesmo trecho. Vamos a um exemplo:

Código do LaTeX:

```
Estou impressionado com a \emph{flexibilidade} do LaTeX.
```

Resultado final:

```
Estou impressionado com a flexibilidade do LaTeX.
```

Em um editor convencional, se colocarmos toda a frase em itálico, ele tentará adivinhar se

desejamos manter a ênfase do termo “flexibilidade” ou se ele deve deixar tudo em itálico. Ocasionalmente, ele vai coincidir com seu objetivo, mas nas outras vezes ele lhe forçará a corrigir novamente o que deve acontecer com o “flexibilidade”. No LaTeX, você não terá dúvida de que a presença do “\emph” significa que aquele termo estará sempre enfatizado *de acordo* com o contexto do documento. Portanto, se eu quiser colocar toda a frase em itálico, eu preciso apenas colocá-la dentro de um *novo* comando de ênfase e o resultado final irá preservar a ênfase inicial.

Vamos voltar ao exemplo anterior. Simplesmente adicionamos um *novo* comando para toda a frase, mantendo a ênfase anterior:

Código do LaTeX:

```
\emph{Estou impressionado com a \emph{flexibilidade} do LaTeX.}
```

Resultado final:

*Estou impressionado com a flexibilidade do LaTeX.*

Embora tenhamos nos detido longamente em um único exemplo de comando, espero que o leitor perceba que o objetivo aqui é compreender as características fundamentais presentes em *qualquer comando* do LaTeX. Cada código pontua um elemento lógico do texto e irá sempre se manter estável ao longo da edição. Para praticamente qualquer necessidade existem várias opções de comandos possíveis para resolver seus problemas.

Por fim, vamos revisar os pontos fundamentais desta seção: 1. Os principais comandos do LaTeX seguem sempre a mesma estrutura básica: \nome-do-comando[variável secundária]{variável principal}). 2. Não é necessário se preocupar em decorar todos os comandos. Você naturalmente irá se lembrar dos mais utilizados e encontrará facilmente os demais por meio de atalhos no editor ou por buscas na internet. 3. Caso lhe incomode ver seu texto misturado com comandos, os editores possuem vários recursos para tornar isso mais tolerável. 4. Esses comandos trarão uma série de vantagens ao seu texto: a. Ele se tornará mais estável, pois os comandos são persistente (continuam funcionando independentemente do resto do documento); b. Ele se tornará mais flexível, pois o comportamento dos comandos se adaptam ao contexto e às definições globais do documento; c. Você terá mais controle sobre documento, pois saberá exatamente o que está ocorrendo com ele.

## A estrutura do documento: o conceito de preâmbulo

A segunda grande diferença entre um documento em LaTeX e um documento convencional é em relação à sua *estrutura*, isto é, em relação a como seus elementos internos estão organizados.

Em um documento convencional, o início do documento é justamente *o próprio documento*. Em outras palavras, a primeira coisa que enxerga em um editor convencional é a primeira página do documento, enquanto a última coisa que você encontra é a última página do documento. No LaTeX, as coisas são um pouco diferente. Como há uma separação entre *aparência* e *conteúdo*, há também uma separação entre a seção onde você determina as características gerais do seu texto e o seu próprio texto.

Nesse sentido, a principal característica da *estrutura* de um documento do LaTeX é que ele pode ser dividido em duas grandes partes: o *preâmbulo* e documento propriamente dito.

O *preâmbulo* serve para você dizer para o LaTeX como interpretar seu documento. No preâmbulo você determina todas as características relativas à aparência do seu documento: as margens, a fonte, as características dos títulos das seções, o tipo de nota de rodapé, a numeração das páginas, etc. Em suma, o *preâmbulo* define as características globais do texto. Evidentemente, não é necessário definir tudo nos mínimos detalhes; o LaTeX geralmente tem bom-senso o suficiente para presumir definições apropriadas para cada caso.

No entanto, o LaTeX exige que haja pelo menos um tipo de informação básica em seu preâmbulo: que você diga para ele que *tipo* de documento que você pretende escrever. Isso é definido pelo comando “\documentclass{tipo-do-documento}”. Algumas classes bastante úteis são “article” (para artigos acadêmicos), “memoir” (para teses acadêmicas) e “book” (para livros). É preciso também citar uma classe especial, desenvolvida pela equipe do abnTeX2 que automaticamente coloca todo o documento nas normas da ABNT (a classe “abntex2”). Porém, dependendo da sua distribuição do LaTeX, essa classe precisa ser instalada separadamente.

No preâmbulo, também podemos adicionar uma série de “pacotes” (*packages*) ao nosso documento. “Pacotes” são espécies de programas auxiliares que adicionam várias funcionalidades ao seu documento. Os pacotes são adicionados pelo comando “\usepackage{nome-do-pacote}”. Em geral, esses pacotes são opcionais, mas para nós há um pacote muito especial: o “xltextra”. Esse pacote é, na verdade, um conjunto de pacotes que ensinam ao LaTeX como lidar com as fontes do documento. Embora ele possua várias funções bastante sofisticadas, por enquanto apenas uma nos interessa: ele irá avisar ao XeLaTeX como interpretar os acentos latinos – e outros alfabetos mais exóticos – que porventura iremos utilizar.

Portanto, enquanto em um editor convencional, você começa já escrevendo seu texto, no LaTeX você deve começar escrevendo seu preâmbulo. Esse preâmbulo não precisa ser rescrito a cada novo arquivo – afinal, uma das vantagens do LaTeX é poder reaproveitar o trabalho já realizado; seja feito por nós mesmo, seja disponibilizado na internet enquanto algum modelo previamente preparado.

Em todo caso, as primeiras linhas do LaTeX sempre iriam constituir o preâmbulo. No próximo capítulo, iremos discutir em mais detalhes o que pode ser colocado nesta seção. Por enquanto, vamos deixá-lo com um exemplo das primeiras linhas de um arquivo em LaTeX.

Prestem atenção em um convenção que utilizaremos ao longo de todo o livro: tudo que está escrito *após* o comando “%” não é necessário para o LaTeX. Esse comando diz ao LaTeX para ignorar o restante da linha. Nós iremos utilizá-lo para introduzir comentários aos códigos que colocamos com exemplo no livro.

Um preâmbulo mínimo:

```
\documentclass{article} % Diz para o LaTeX que iremos escrever um documento da classe "article".  
\usepackage{xltextra} % Diz para o LaTeX carregar o pacote "xltextra" que o ensina a lidar com a codificação do nosso texto.
```

## A estrutura do documento: o texto

O LaTeX sempre vai presumir que seu documento *começa* com o preâmbulo. Portanto, ele vai “ler” tudo que você escrever nas primeiras linhas do seu documento como sendo parte do preâmbulo até o momento em que você avisa que seu texto propriamente dito irá começar. Portanto, se você já tentou criar um documento em LaTeX antes de ler essa parte a apareceu algum erro é justamente por isso: o LaTeX pensa que as primeiras linhas são códigos do preâmbulo e não exatamente o seu texto.

Portanto, é necessário *avisar* ao LaTeX que seu preâmbulo terminou e que texto começou. Isso é feito por um comando que avisa ao LaTeX que foi criado um novo “ambiente” que terá regras próprias. Esse comando segue as características gerais que vimos acima como uma pequena diferença: todo comando que cria um *ambiente* precisa também avisar que o ambiente terminou. Portanto, você precisa dizer ao LaTeX tanto que seu texto *começou* quanto avisá-lo quando ele irá terminar. O comando para fazer isso é bastante simples: você utiliza o `\begin{document}` para assinalar o começo do texto e o comando `\end{document}` para avisar que texto terminou.

Podemos ver que esse comando segue as características que já estudamos, com a diferença de possuir um *começo* e um *fim*: temos a barra invertida, o nome do comando (“begin”) e a variável (no caso, é “document”, que é a variável utilizada para indicar o conteúdo do texto). O comando de encerramento segue a mesma lógica apenas trocando o “começo” (begin) por “fim” (end). Como iremos reencontrar essa mesma estrutura em diferentes comandos, é interessante compreender suas características. Sempre que um comando possui os pares “begin” e “end”, trata-se de um comando do LaTeX para criar um “ambiente”, isto é, uma seção do documento que possui regras especiais.

O LaTeX exige que todo documento tenha ao menos um único grande ambiente chamado “document”: todo o corpo do seu texto estará dentro desta seção. Entretanto, *dentro* desse ambiente maior, podemos criar inúmeros ambientes especiais, com regras próprias. Por exemplo, para incluir uma citação em nosso texto (que exige regras próprias para margem, fonte e recuo de parágrafo), também utilizamos um comando análogo: “`\begin{quote}`” e `\end{quote}`”. Em suma: embora existam vários comandos para as mesmas coisas, eles seguem uma lógica muito parecida, o que facilita sua memorização e utilização.

Resumindo, no LaTeX, todo arquivo deve possuir dois elementos principais: (i) o preâmbulo, que *começa* na primeira linha e vai até o momento em que você “avisa” ao LaTeX que seu texto *começou*; (ii) o texto propriamente dito, que é colocado *entre* os dois comandos “`\begin{document}`” e “`\end{document}`”. O seu texto deve vir inteiramente escrito *entre* esses dois comandos que assinalam o “ambiente” onde está o seu documento.

Estamos agora em condições de produzir um primeiro arquivo básico de LaTeX. Se o leitor salvar o seguinte código e salvá-lo como um arquivo “.tex”, ele terá seu primeiro arquivo válido em LaTeX.

Vamos observar o código comentado do arquivo “lutex01.tex”

```
\documentclass{article} % Elemento necessário do preâmbulo que indica a utilização da classe "article".  
\usepackage{xltextra} % Elemento opcional do preâmbulo que carrega o pacote "xltextra".  
\begin{document} % Avisa que o "ambiente" document começou.  
O \emph{conteúdo} do seu documento fica aqui. % Tudo que você escrever a partir dessa linha  
fará parte do seu documento.  
\end{document} % Avisa que o conteúdo do seu documento terminou. É importante para que o  
LaTeX funcione normalmente.
```

Sei que parece muito trabalho para um texto simples, mas peço que o leitor se lembre da nossa máxima: o LaTeX é difícil no começo, mas fica progressivamente mais fácil com o tempo; o leitor logo verá que funções extremamente complexas serão realizadas a partir de modificações mínimas no arquivo acima. Compreender o documento acima talvez seja a etapa mais difícil no aprendizado do LaTeX.

## Falhando no LaTeX: lidando com erros

Com os elementos expostos anteriormente, o leitor já terá todos os elementos para produzir seu primeiro arquivo em LaTeX. Recomendamos que, a partir desse momento, o leitor comece a fazer experimentos, tentando replicar os exemplos apresentados ao longo do livro.

Antes de ir para a prática, no entanto, preciso lhe preparar para algo que talvez seja uma surpresa: para ser sincero, o processo de transformações dos arquivos do LaTeX para o formato PDF é um pouco esquisito. Parte dessa esquisitice envolve aprender a lidar com as mensagens de erro que serão geradas pelo LaTeX. Sem o menor exagero, eu diria que aprender a lidar com os erros é o aspecto mais importante do aprendizado do LaTeX.

Em um editor convencional, as mensagens de erros significam que o programa travou e que é preciso reiniciá-lo. São, portanto, falhas catastróficas que interrompem todo o trabalho. O LaTeX raramente tem erros desse tipo – e, na verdade, isso nunca aconteceu na minha experiência. Como o sistema é muito antigo e estável (e simples), esse tipo de erro não tende a ocorrer.

No entanto, o que ocorre com enorme frequência é outro tipo de erro: quando o LaTeX não consegue interpretar o código que você escreveu. Esse não é um erro do sistema, mas um erro na formatação do seu arquivo. É apenas um sinal de que é preciso revisar seu documento para entender o que houve de errado. Como estamos acostumado a pensar nos erros como falhas catastróficas, tendemos também a nos exasperar com os erros do LaTeX. No entanto, é preciso

fazer o esforço de nos lembrar que eles não são tão sérios: são apenas problemas de formatação que, com alguma atenção (e com a ajuda do Google), serão facilmente resolvidos. Ao invés de se irritar e desistir com a primeira (e a milésima) mensagem de erro, você precisa desenvolver estratégias para solucioná-las.

Para fazer isso, o primeiro passo é compreender a razão desses erros. Eles estão intrinsecamente relacionados ao modo como o LaTeX transforma seus arquivos “.tex” em arquivos “.pdf”. Novamente, quem é da área de computação talvez ache esse processo quase banal, mas a minha experiência é que o pessoal da área de humanas tende a se assustar com a possibilidade de um programa que “ainda” pode rodar diretamente do *prompt de comando*. Felizmente, os editores de LaTeX diminuem essa estranheza ativando o TeX diretamente. No entanto, essa ativação possui dois efeitos colaterais: a geração de arquivos temporários e as mensagens de erro.

Como já foi dito anteriormente, uma coisa é *escrever* um artigo em LaTeX e outra coisa é *transformá-lo* em um “arquivo pdf”. Essa transformação é feita por um sistema tipográfico que começa a “ler” seu arquivo original e busca interpretar a melhor forma de gerar a apresentação final. Se tudo ocorrer bem, seu arquivo “exemplo.tex” irá gerar um novo arquivo chamado “exemplo.pdf”: legível em qualquer computador e exatamente dentro da formatação desejada. No entanto, *nem sempre as coisas ocorrem bem*.

Durante esse processo, o sistema tipográfico do LaTeX pode encontrar – e geralmente encontra – algum tipo de erro. Se for um erro muito pequeno, ele irá continuar e gerar o arquivo assim mesmo. Se for um erro crítico, ele irá interromper a geração do arquivo e esperar que você corrija o erro. Em geral, no entanto, mesmo em relação aos pequenos erros, é importante descobrir o que aconteceu para deixar seu arquivo exatamente do jeito desejado.

Há uma imensa variedade de erros possíveis. Pode ser desde um erro de digitação (“\emhp” no lugar de “\emph”) ou, em casos mais complexos, a utilização de pacotes incompatíveis. Em todos os casos, o LaTeX tentará lhe ajudar e irá apresentar uma mensagem de erro identificando o problema. Em geral, ele tenta apontar diretamente para o número da linha onde o código gerou o problema. No início, no entanto, a explicação do LaTeX parece muito complicada, mas aí é que entra a magia da internet: pode ter certeza que alguém já cometeu o mesmo erro. Como o LaTeX é amplamente utilizado no mundo inteiro, a comunidade virtual é bastante ativa. Se você não entendeu a mensagem de erro publicada pelo LaTeX, você certamente encontrará fóruns de discussão onde alguém pediu ajuda em uma situação bastante parecida com a sua.

Portanto, para trabalhar com o LaTeX, é importantíssimo pertermos o medo das mensagens de erro e começarmos a aprender a interpretá-las. Se mandarmos nosso editor gerar um arquivo “.pdf” e nada acontecer, ele provavelmente vai imprimir uma longa lista de todos os processos internos realizados. Ao final dessa lista, geralmente haverá uma mensagem de erro com o nome do arquivo e o número da linha problemática. Voltando ao seu arquivo, você provavelmente encontrará o erro que causou o problema. Se isso não for suficiente, uma busca na internet deverá trazer a solução.

Bom, mas mesmo quando tudo ocorre bem, o LaTeX ainda irá gerar uma série de artigos auxiliares para gerar o documento final. Portanto, se *tudo ocorrer bem*, além dos arquivos “exemplo.tex” e “exemplo.pdf” você vai ter também uma série de arquivos intermediários chamados “exemplo.aux”, “exemplo.toc”, “exemplo.log”, etc. Esses arquivos são necessários apenas *durante* o processo de geração do “exemplo.pdf”, mas o sistema do LaTeX não os apaga automaticamente, pois eles contém algumas informações que podem ser importantes para

entender o que aconteceu na geração tipográfica do seu arquivo. Portanto, se você não precisar verificá-los para compreender algum problema, você pode apagar todos os arquivos temporários gerados pelo LaTeX – mas *apenas depois* que seu arquivo final estiver pronto, .

Embora talvez seja óbvio, acho bom avisar que é importante ter o cuidado em manter sempre o arquivo “exemplo.tex” original depois que apagar os arquivos temporários. É importante compreender que a transformação do arquivo “.tex” em arquivo “.pdf” é uma via de mão única: depois que você estiver com o “exemplo.pdf” em mãos, poderá mostrá-lo para quem quiser; para fazer modificações, no entanto, é preciso sempre voltar diretamente ao “exemplo.tex”. Portanto, depois de terminado o processo, sinta-se à vontade para apagar os arquivos temporários, mas mantenha o arquivo “tex” original: ele é seu objeto de trabalho.

Por esses motivos, eu acho bastante prático manter cada projeto que faço no LaTeX em seu próprio diretório. Por exemplo, quando escrevo um novo artigo, crio uma pasta exclusivamente para ele e um novo “arquivo.tex” (posteriormente, mostrarei um comando que me poupa o trabalho em escrever um novo preâmbulo a cada vez). Desse modo, fica mais fácil apagar os arquivos temporários criados durante o processo sem arriscar apagar nada importante.

## Vencendo no LaTeX: o primeiro arquivo

Nesse ponto, já sabemos o suficiente para criar um arquivo LaTeX. Por isso, a partir de agora, espero que os leitores comecem ativamente fazer seus experimentos enquanto acompanham a leitura do livro.

Antes disso, no entanto, vamos fazer uma revisão geral para ver tudo que precisamos para ter um arquivo funcional do LaTeX:

1. O início do documento é o *preâmbulo*: ele dá instruções ao LaTeX sobre *como interpretar* o documento. O preâmbulo define o *tipo* de documento e define pacotes adicionais para ampliar as funcionalidades do LaTeX.
2. Vimos também que é necessário escolher uma “classe” para seu documento. Exemplos de classes possíveis: article, memoir, book. Vamos começar utilizando a classe “article” por ser mais simples.
3. O preâmbulo também ensina ao LaTeX como utilizar fontes. No nosso caso, é importante utilizar um pacote que o ensina a ler os caracteres latinos. O pacote “xltextra” possui vários recursos; entre eles, um pacote que vai garantir que nossos acentos estarão corretos na versão final.
4. Para assinalar que o preâmbulo terminou e que o texto começou, utilizamos o comando “\begin{document}”. Precisamos também assinalar o final do nosso texto com o comando complementar “\end{document}”.
5. Aprendemos que os comandos seguem o modelo “\nome-do-comando{variável}”.
6. Aprendemos também que podemos colocar um trecho em itálico com o comando “\emph{trecho em itálico}”.
7. Sabemos também que, *depois de escrever o arquivo*, precisamos *transformá-lo* em um “arquivo.pdf” para que possa ser lido por outras pessoas.

8. Nesse processo, se ocorrer algum erro, o LaTeX irá nos avisar o que aconteceu. Se não entendermos a mensagem do LaTeX, podemos buscar por esse erro na internet onde certamente encontraremos pessoas que passaram por experiências semelhantes.
9. Mesmo se não houver erros, o LaTeX irá gerar vários arquivos auxiliares. Podemos deletá-los *depois* que o processo terminar, isto é, depois que o arquivo “.pdf” tiver sido criado. Para facilitar a exclusão dos arquivos temporário e diminuir o risco de apagar os arquivos principais, recomendamos criar uma pasta para cada projeto no LaTeX.
10. Lembre-se que precisamos utilizar um sistema tipográfico específico para preservar a formatação do texto: o *XeLaTeX*. No TeXworks e no TeXShop você pode escolher o sistema desejado a partir de uma lista na barra de atalhos.

Juntando tudo que foi dito anteriormente, recomendamos que o leitor crie um novo diretório para escrever seu primeiro artigo em LaTeX. Lembre-se de salvar o artigo utilizando o formato de codificação UTF-8 (o que é feito automaticamente no TeXworks e no Sublime Text) e salvar seu primeiro documento com a terminação “.tex”. Depois disso, você terá algo muito próximo do exemplo abaixo e, utilizando o XeLaTeX, poderá gerar um documento com o texto contido dentro do ambiente “document”.

Com posse dessas afirmações, recomendamos que o leitor faça experimentos em torno do exemplo da seção anterior que vamos repetir logo abaixo. Dessa vez, no entanto, vamos sugerir que o leitor cometa alguns erros intencionalmente, para aprender a interpretar as mensagens de erro do LaTeX.

Novamente, lembre-se que o código “%” *não* será lido pelo LaTeX. Esse código diz para o LaTeX ignorar o restante da linha. É um modo dos autores inserirem comentários que serão lidos apenas por quem tiver acesso ao código-fonte do documento. No trecho abaixo, vamos utilizá-lo para propor exercícios. da linha do arquivo.

```
\documentclass{article} % Teste outros comandos e experimente apagar essa linha. \usepackage{xltextra} % Essa linha é opcional, então o LaTeX será capaz de rodar o arquivo sem ela. No entanto, os acentos não serão transcritos corretamente.

\begin{document} % Outro elemento necessário: avisa que o texto do seu arquivo começou.

O \emph{conteúdo} do seu documento fica aqui. % O que acontece se o comando “\emph” for escrito errado?

\end{document} % Outro elemento necessário: avisa que o documento acabou.
```

Se você salvar esse arquivo como “exemplo.tex” exatamente como está e tentar rodá-lo pelo XeLaTeX, terá um arquivo chamado “exemplo.pdf” e vários arquivos temporários com nomes semelhantes (exemplo.aux, exemplo.log, etc.). Vamos discutir algumas mensagens de erro que podem aparecer.

Vamos começar apagando a linha da classe do documento e tentar rodá-lo. No meu programa,

o LaTeX lança o seguinte aviso: “Error: \usepackage before \documentclass”. Embora seja uma mensagem um pouco obscura, sabemos o que aconteceu de errado: sem o comando “\documentclass” o LaTeX não sabe que tipo de documento é este, então ele não sabe como processar o resto do arquivo. No caso, ele reclama que tem que lidar com um pacote sem saber que tipo de documento é esse (e, portanto, sem saber como utilizar o pacote). Em outras palavras: não podemos esquecer de escolher uma classe para nossos documentos!

E se deixarmos a classe, mas retirarmos o pacote de fontes? Dessa vez, o LaTeX não acusa nenhum erro, pois os pacotes são opcionais. No entanto, quando olharmos para o arquivo final, veremos que os acentos sumiram. Melhor colocar o “xltxt” de volta!

Vamos fazer um último experimento: vamos tentar rodar esse arquivo por outro sistema tipográfico. Por exemplo, vamos trocar a opção “XeLaTeX” pela opção “pdfLaTeX” e ver o que acontece. Nesse caso, aparece uma mensagem bem mais longa com um nota explicativa: “XeTeX is required to compile this document. Sorry!”. O motivo desse erro é que estamos utilizando um pacote (o xltxt) que exige o LaTeX. E se retirarmos o pacote e utilizar o pdfLaTeX novamente? Nesse caso, o programa roda sem problemas, mas... sem o “xltxt” e sem o XeLaTeX os acentos latinos (e vários outros caracteres especiais) não aparecem corretamente. Melhor voltar para o XeLaTeX!

Depois desses experimentos, o leitor pode se sentir à vontade para apagar os arquivos temporários que foram gerados. Como os arquivos são bastante pequenos, não é absolutamente necessários apagá-los. No entanto, depois de modificar o arquivo “.tex” original e gerar um novo arquivo “.pdf” é interessante apagar os arquivos temporários: normalmente, o LaTeX é capaz de reescrevê-lo, mas ocasionalmente a presença de arquivos temporários antigos gera alguns problemas de compatibilidade. Evite apenas apagar os arquivos temporários *durante* o processo de geração do arquivo PDF!

## Os próximos passos

Novamente, entendo que parece esforço demais para produzir um arquivo tão simples. No entanto, o ponto principal desse capítulo foi compreender *como* lidar com os arquivos em LaTeX. Veremos que os arquivos mais complexos serão inteiramente produzidos em cima dos mesmos princípios discutidos neste capítulo: com apenas mais algumas linhas de código, documentos elaboradíssimos poderão ser produzidos.

Nesse ponto, espero que o leitor saiba identificar os elementos do *préambulo* (a classe do documento e os pacotes utilizados), saiba avisar ao LaTeX onde o documento começa e que esteja preparado para lidar com as pequenas idiossincrasias do LaTeX (as mensagens de erros e arquivos temporários).

Rigorosamente falando, os procedimentos explicados até esse momento podem ser extrapolados – com a ajuda de guias e fóruns de discussão online – para produzir os textos mais elaborados possíveis. O problema, evidentemente, é decidir o *grau* de elaboração desejado. O LaTeX é tão poderoso e flexível que sempre há um novo “pacote” para adicionar funcionalidades. Por isso, ninguém é capaz de conhecer todas as possibilidades oferecidas por ele. É preferível adquirir uma noção sólida dos fundamentos e ir progressivamente estudando os pacotes e as funcionalidades que lhe interessam para resolver os casos especiais no momento em que eles se apresentam.

Creio que o modo mais eficiente de progredir com o LaTeX é adotar uma postura de curiosidade, aprendizagem constante e solução de problemas. Ao invés de procurar entender absolutamente tudo sobre os vários programas relacionados, é mais interessante apenas adquirir um bom domínio dos fundamentos do LaTeX e ir progressivamente procurando soluções para cada caso específico quando for necessário produzir um documento com características particulares.

O restante do livro tem como objetivo justamente oferecer esses fundamentos básicos. Não seria de modo algum possível oferecer um tratamento exaustivo de todas as funcionalidades possíveis oferecidas pelo LaTeX. Lembrando que o público-alvo do livro é quem pretende utilizar o LaTeX prioritariamente como instrumento para *produzir textos*, deixaremos de lado funcionalidades específicas para matemáticos e programadores – temos certeza que, com os fundamentos adquiridos nesse livro, será fácil para eles encontrarem soluções específicas para seus problemas nos vários outros guias disponíveis na internet.

Dito isso, o próximo capítulo tentará explicar os fundamentos básicos sobre como o LaTeX interpreta os textos (o que provavelmente será de interesse universal) e depois discutiremos questões que talvez interessem um público mais específico (administração de bibliografia e preparação de apresentações).

# Capítulo 4: Como o LaTeX lida com o texto

## Introdução

Nos capítulos anteriores, vimos que o LaTeX promove uma separação entre a *aparência* e o *conteúdo* do texto por meio de uma combinação entre *texto puro* e *códigos de formatação*. Evidentemente, isso deixa implícito um problema que precisamos aprofundar: qual é exatamente a relação entre os códigos e o texto propriamente dito? Como o LaTeX decide *como* interpretar os diferentes sinais gráficos que compõe um texto?

Preciso reconhecer que, por motivos didáticos, recorri à uma noção bastante simplista de “*texto puro*”. O texto mais simples possível é composto por frases, sinais de pontuação e parágrafo. Temos ainda uma série de sinais gráficos especiais que o LaTeX interpreta como códigos de formatação, dos quais já descobrimos dois: a barra invertida () e o sinal de porcentagem (%).

Nesse capítulo, iremos discutir as convenções textuais presumidas pelo LaTeX. Muitas dessas convenções podem ser modificadas no *preâmbulo*, mas é interesse conhecer o comportamento padrão do LaTeX antes de realizar modificações. Algumas dessas convenções são particularmente importante porque elas irão modificar o modo como *escrevemos* nossos textos.

## O conceito de linha no LaTeX

A primeira grande diferença entre escrever textos no LaTeX e escrevê-los em um editor convencional diz respeito à noção de linha. Novamente, esse é um assunto que os programadores compreendem instintivamente, mas que passa despercebido por quem vem da área de humanas.

Para nós, que crescemos com editores de textos convencionais, uma linha de texto continua até o limite da tela do computador. Portanto, assim, que a gente aumenta o tamanho da fonte ou a definição do monitor, nosso texto subitamente se adapta às novas configurações visuais. No LaTeX, no entanto, a noção de linha possui *limites lógicos* e não *limites visuais* – e é muito importante compreender essa distinção.

Por “limite visual” da linha, estou me referindo à decisão do seu editor de texto em dividir uma sequência longa de texto em várias linhas para facilitar sua visualização na tela do computador. Esse limite é puramente acidental e irá mudar constantemente se você mudar o tamanho da tela ou a definição do monitor. Por “limite lógico” estou me referindo à inserção de um comando específico para assinalar que aquela linha chegou ao fim e que, independentemente de quanto espaço possuímos na tela ou no documento, o editor deve iniciar uma linha inteiramente nova. Esse limite é constante, pois é voluntariamente delimitado; independentemente das características do nosso monitor, ele permanece o mesmo.

Em editores convencionais essa distinção é difícil de perceber, pois não temos acesso direto aos códigos por trás do documento. E, mesmo em alguns editores de LaTeX, essa distinção não é enfatizada. Porém, como essa distinção é muito importante para o LaTeX (e para programadores em geral), ela aparece em editores mais poderosos. Essa distinção ficará bastante clara ao observarmos a diferença entre as imagens de dois arquivos produzido no Sublime Text.

Vamos à primeira:

```

1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5 Esse documento tem apenas uma única linha em
termos lógico. No entanto, ela é tão longa que
termina por forçar o editor a dividir visualmente
o texto em quatro linhas.
6 \end{document}

```

Arquivo “lutex02.tex” com fonte maior

Neste primeiro caso, temos um documento com uma longa frase que o editor, por conveniência, dividiu *visualmente* em várias linhas. Note-se que, em nenhum momento em apartei *enter* ou *return* para quebrar a linha; o editor fez essa divisão apenas para manter o texto inteiramente visível na tela. No entanto, apesar de *visualmente* o texto estar “quebrado”, em *termos lógicos*, todo o texto continua em única linha. Para compreender essa distinção, prestem atenção às diferentes linhas e os números que a antecedem neste exemplo.

O Sublime Text deixa essa distinção bastante clara ao *enumerar* as linhas do texto. Se olharmos para a extremidade esquerda da imagem, veremos uma série de números que designa a *ordem lógica* das linhas do texto. Vemos que, na primeira linha, temos o número “1” seguido de um código do LaTeX e o mesmo ocorre na segunda e na quarta linha. Na terceira linha, no entanto, já temos algo diferente: temos o número “3” seguido de uma linha em branco. Isso significa que, embora não haja texto nessa linha, nós *criamos* esse espaço vazio voluntariamente em nosso editor; ou seja, do ponto de vista da estrutura lógica do documento, a terceira linha existe mesmo estando vazia.

Em todo caso, o mais interessante dessa imagem é o que ocorre a partir da quinta linha: temos o número “5” seguido de uma sequência de texto que continua por várias linhas. No entanto, o Sublime Text *não enumera* as linhas seguintes. Isso ocorre porque em nenhum momento foi

dado o comando para criar uma “quebra de linha” nessa passagem. Por isso, embora *visualmente* a quinta linha do documento continue por *quatro* linhas, *logicamente* todo esse texto ainda faz parte exclusivamente da quinta linha. Por isso, o Sublime Text só volta a enumerar as linhas a partir do trecho onde temos o código de encerramento do documento (`\end{document}`) que aparece na sexta linha em termos lógico (embora, visualmente, já seja a décima linha do texto).

Para ficar mais claro, vamos pegar *exatamente* o mesmo arquivo e ver o que acontece quando modificamos a configuração da tela para caber mais texto na mesma imagem.

```

1 |\documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5 Esse documento tem apenas uma única linha em termos lógicos. No entanto, ela é tão longa que termina por forçar o editor a dividir visualmente o texto em quatro linhas.
6 \end{document}

```

Arquivo “lutex02.tex” com fonte menor

Note-se que não houve nenhuma modificação no *conteúdo* do arquivo; a única coisa que mudou foi a definição da tela. Por isso, o editor de texto teve mais espaço para rearranjar o conteúdo do arquivo e colocá-lo em uma única linha. Note-se, no entanto, que a numeração das linhas permanecesse estritamente igual: todo o texto *continua* na *linha 5*, o documento *continua* terminando na *linha 6* e a *linha 3* continua em branco. Em outras palavras: embora visualmente o documento esteja diferente, a estrutura lógica do documento permanece exatamente a mesma. Nesse sentido, o LaTeX considera as linhas como elementos tão importantes da estrutura do texto que ele as trata como elementos persistentes, independentemente dos arranjos visuais particulares.

Embora essa diferença seja sutil, ela é extremamente importante por mostrar o poder e a estabilidade que o LaTeX confere ao usuário sobre a aparência do seu texto. Ao contrário de um editor de texto convencional, não temos surpresa alguma quanto ao que está acontecendo por trás do nosso texto. Sua estrutura interna está logicamente delimitada e permanece estável mesmo com mudanças na aparência do editor: uma linha só será “quebrada” se nós realmente quisermos que ela seja divida e avisarmos ao LaTeX para fazer isso.

Complementarmente, vamos observar um exemplo contrário, onde eu optei deliberadamente

por quebrar as linhas do meu texto. Para fazer isso, apertei *enter* (ou *return*, dependendo do seu teclado) ao final de cada frase que eu queria separar. Apesar das frases serem curtas, esse comando força o LaTeX a considerar que elas agora fazem partes de linhas *logicamente* distintas. Vejamos como isso ocorre na prática:

```
1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5 Uma linha com um única frase.
6 Outra linha com uma única frase.
7
8 A linha anterior ficou em branco!
9 \end{document}
```

Line 8, Column 34      Tab Size: 4      LaTeX

Arquivo “lutex03.tex” com fonte maior

Como podemos ver, a numeração das linhas desse novo artigo é inteiramente diferente. As quatro primeiras linhas são absolutamente iguais, mas as coisas mudam a partir da quinta linha. Embora essas linhas sejam mais curtas do que as linhas do arquivo anterior, dessa vez eu *intencionalmente* quis separá-las. Portanto, como agora há uma separação lógica entre essas linhas, o meu editor (o Sublime Text) agora as enumera de modo diferente. As linhas seguintes recebem progressivamente os números seguintes, de modo que a última linha agora possui o número 9. Em suma, embora haja a mesma quantidade de texto, no mesmo espaço visual, o LaTeX reconhece que algo diferente está acontecendo.

Note-se também que a estrutura lógica também permanece inalterada se alterarmos as características visuais da imagem. Vamos observar o que acontece com o arquivo anterior com mais espaço em tela:

```

1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5 Uma linha com um única frase.
6 Outra linha com uma única frase.
7
8 A linha anterior ficou em branco!
9 \end{document}

```

Arquivo “lutex03.tex” com fonte menor

Vemos que a numeração também permanecesse inalterada neste exemplo. Mesmo havendo um espaço quase exagerado para manter todo o texto na mesma alinha, *ainda assim* o LaTeX mantém a separação desejada. Em suma, a noção de linha no LaTeX é tratada de modo constante ao longo do documento. Isso ocorre porque as quebras de linha possuem um valor *lógico* e não *visual*. O objetivo dessa distinção é fornecer mais estabilidade e controle sobre nossos documentos. Estritamente falando, a divisão das linhas é equivalente aos comandos textuais do LaTeX, pois indicam como ele deve interpretar o texto.

Em outras palavras: ao redigir um documento, você precisa saber que o LaTeX só irá considerar que você iniciou uma nova linha se você *ativamente* avisar que deseja fazê-lo, apertando *enter* para “quebrar” a linha. O motivo pelo qual o LaTeX leva as linhas tão à sério é porque isso possui uma enorme influência sobre o modo como ele interpreta o texto – o que veremos na seção a seguir.

## Como o LaTeX interpreta as linhas do seu documento

Acabamos de ver que o LaTeX possui um conceito de linha bastante específico: ele simplesmente pressupõe que todo texto continua indefinidamente até o momento em que o usuário *ativamente* avisa que a linha deve ser interrompida por um comando.

Vamos agora retornar os exemplos da seção anterior e ver como o LaTeX gerou cada arquivo. Como vimos, não faz diferença o modo como o arquivo *aparecia* em nossa tela; independentemente da definição visual, ambos os artigos permaneciam com a mesma estrutura lógica. Agora, vamos ver como os dois arquivos visto na seção anterior são gerados pelo LaTeX.

Primeiramente, vamos rever o código fonte do arquivo “lutex02.tex”:

```
\documentclass{article} \usepackage{xltxtra}  
\begin{document} Esse documento tem apenas uma única linha em termos lógico. No entanto,  
ela é tão longa que termina por forçar o editor a dividir visualmente o texto em quatro linhas.  
\end{document}
```

Depois de rodá-lo pelo XeLaTeX, vemos que o seguinte arquivo (lutex02.pdf) será gerado:

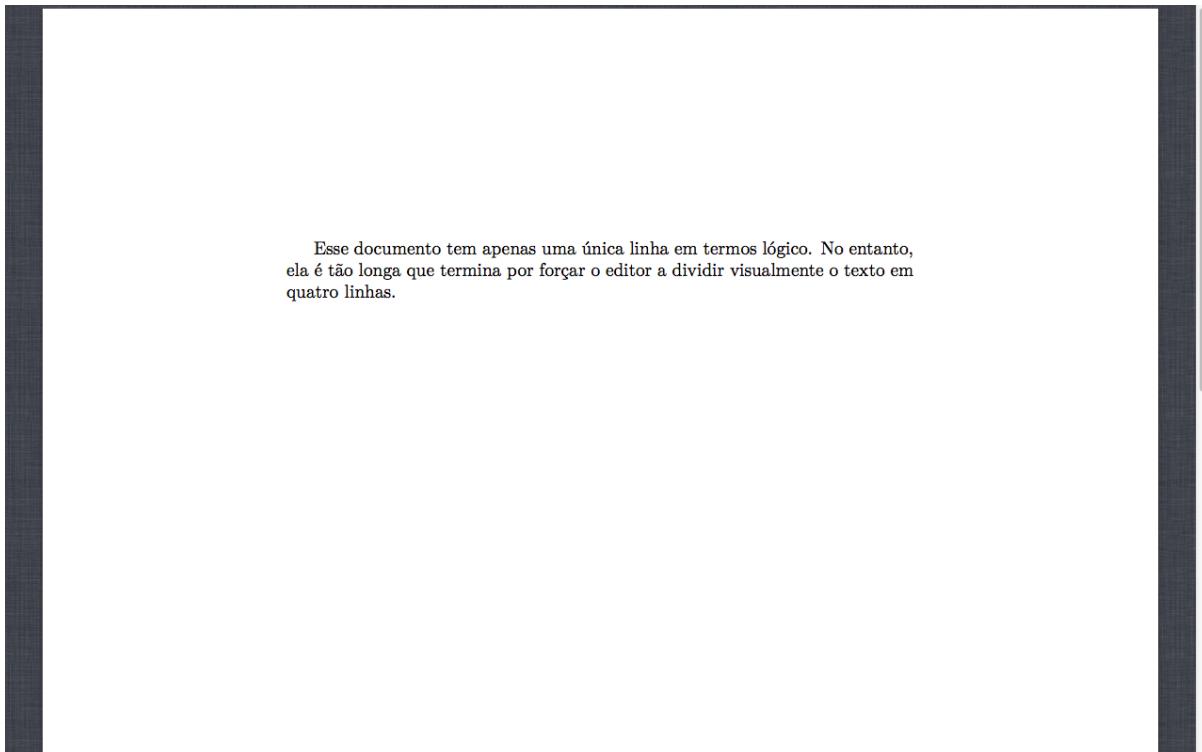


Imagen do arquivo “lutex02.pdf”

Olhando para o exemplo, vemos que algo curioso aconteceu: o texto não aparece em um única linha nem aparece nas quatro linhas como estava em nosso editor de texto. O que aconteceu, na verdade, foi algo bastante simples e sensato: como todo nosso texto estava em única linha, o LaTeX *entendeu* que todo aquele texto fazia parte do mesmo parágrafo e fez o possível para colocá-lo do modo mais visualmente agradável de acordo com o tipo de documento assinalado no preâmbulo. Como o LaTeX é um sistema tipográfico muito inteligente, não precisamos pensar no espaço que nosso texto está ocupando no editor, pois ele se encarregará sozinho de publicá-lo da melhor forma possível de acordo com o espaço das margens e o tipo de papel do documento final<sup>4</sup>.

<sup>4</sup>Evidentemente, é possível *forçar* o LaTeX a modificar seu comportamento convencional por meio de códigos adicionais, mas é preferível fazer isso apenas quando for absolutamente necessário. Para quase todos os casos, o LaTeX é inteligente o suficiente para organizar o texto do melhor modo possível.

Em outras palavras, não importa a *aparência* do texto no editor; o que importa é a *estrutura lógica* do texto em nosso editor. Ao colocarmos todo aquele trecho em uma linha, nós dissemos ao LaTeX que todo esse texto deveria estar no mesmo parágrafo. Ao dizermos que o documento era da classe “article” e nos omitirmos em dizer o tamanho da fonte, o tipo de papel e o tamanho das margens, o LaTeX se encarrega de calcular automaticamente o valor ideal para cada um desses fatores e organizar o texto do melhor modo possível – se quiséssemos utilizar valores diferentes, bastaria informar ao LaTeX esses valores no preâmbulo.

Em suma, para o LaTeX, a noção de linha está relacionada com a unidade lógica do parágrafo: ao ver um texto em única linha, o LaTeX fará o possível para mantê-lo no mesmo parágrafo. Os limites do parágrafo, por sua vez, serão calculados automaticamente a partir das características globais do documento explícitas (ou implícitas) no preâmbulo.

Passemos agora a um novo exemplo. Vamos relembrar o código-fonte do segundo exemplo da seção anterior:

```
\documentclass{article} \usepackage{xltxtra}  
\begin{document} Uma linha com um única frase. Outra linha com uma única frase.  
A linha anterior ficou em branco! \end{document}
```

Faremos agora o mesmo procedimento e vamos rodar o arquivo pelo XeLaTeX para vermos como será o arquivo final gerado.



Imagen do arquivo “lutex03.pdf”

Temos agora um resultado bastante curioso. Vemos que, apesar das duas primeiras linhas serem logicamente distintas, o LaTeX as colocou no mesmo parágrafo. Enquanto isso, não há nenhum sinal que indique o que aconteceu com a terceira linha que estava em branco. Enquanto isso, a quarta linha está em um parágrafo separado (podemos ver isso pelo fato de que a segunda linha do texto começa com um recuo de início de parágrafo, o que não ocorreu no exemplo anterior).

Para entendermos o que aconteceu, precisamos entender como o LaTeX interpreta as quebras de linhas *propositais*, isto é, as quebras de linhas intencionalmente demarcadas por um comando do autor do documento. Nesse exemplo, todas as linhas foram divididas logicamente. No entanto, o LaTeX interpretou cada quebra de linha de um modo diferente. Isso acontece por causa de certas convenções textuais adotadas pelo LaTeX para lidar com linhas.

Já vimos uma dessas convenções: o LaTeX considera que todo o texto que está na mesma linha faz parte do mesmo parágrafo. Como ele lida com textos que estão em linhas diferentes?

Existe uma convenção curiosa adotada pelo LaTeX: ele ignora a primeira quebra de linha. Ou seja: para o LaTeX, se o texto estiver separado apenas por uma *única* quebra de linha, ele *ainda* considera que o texto faz parte do mesmo parágrafo. Apenas quando existem *duas* quebras de linha – isto é, quando há uma linha inteira em branco – o LaTeX entende que o autor quer indicar que está na hora de começar um parágrafo diferente.

Embora isso pareça estranho, há uma boa razão por trás disso. O LaTeX considera que a primeira quebra de linha serve apenas para o autor organizar o texto *para si mesmo*, sem qualquer consequência para a apresentação final do texto. Essa divisão equivale à separação entre a quarta e quinta linha do nosso exemplo: como havia apenas uma quebra de linha entre elas, o LaTeX entendeu que deveria continuar no mesmo parágrafo.

Sei que ainda não dissipei completamente a estranheza; o leitor deve estar se perguntando

porque alguém iria querer organizar o próprio texto separando cada frase em uma linha distinta. Há realmente um caso onde isso é útil: caso o autor do texto utilize sistemas de *controle de versão* – como o *git*, uma ferramenta utilizada por programadores – é bastante útil manter cada frase em uma linha separada, pois fica mais fácil identificar as mudanças que ocorrem ao longo das revisões feitas no texto. Além disso, esse sistema torna mais fácil a colaboração com outros autores, pois cada um pode identificar exatamente que linhas foram alteradas pelos outros colaboradores.

Evidentemente, isso não significa de modo algum que *nós* devemos a começar escrever uma única frase por linha. Para nós, que somos acostumados a escrever textos mais longos, esse procedimento seria muito diferente do nosso hábito normal, o que afetaria nosso fluxo de trabalho. Por isso, é perfeitamente aceitável ignorar essa convenção do LaTeX e escrever várias frases na mesma linha. No entanto, *apenas* para aqueles que desejam dividir seu documento de um modo extremamente meticuloso, o LaTeX oferece essa funcionalidade ao ignorar *apenas a primeira quebra de linha*, presumindo que ela foi feita exclusivamente para o autor, sem qualquer efeito sobre o documento final.

No entanto, quando ocorrem *duas quebras de linha* (como ocorre, em nosso exemplo, entre a linha 2 e 4), isto é, *quando uma linha inteira é deixada em branco*, então o LaTeX entende que se trata de um comportamento proposital que deverá afetar o documento final. No caso, a convenção do LaTeX é interpretar que, sempre que houver uma linha em branco, isso significa que o autor deseja iniciar um novo parágrafo. Por isso, ele fará exatamente o que fez no exemplo anterior: seguindo as convenções pré-determinadas no preâmbulo (relativas às margens, ao tamanho da fonte, ao tipo de papel e ao tipo do documento), ele encontrará o melhor modo de organizar visualmente os parágrafos assinalados.

Para tornar isso mais claro, vamos criar um novo exemplo e ver como o LaTeX irá gerar o arquivo final. Vamos ver o que acontece com o arquivo “lutex04.tex”. Para ficar mais claro a relação entre linhas e código, colocarei uma imagem novamente feita a partir do Sublime Text:

```

1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5 Nessa primeira linha, colocaremos várias frases. No entanto, como não iremos não
6 iremos assinalar ao LaTeX nenhuma quebra de linha, todas elas ficarão no mesmo
7 parágrafo.
8
9 Para assinalarmos que estamos em um novo parágrafo, deixamos a linha anterior em
10 branco. Novamente: não precisamos nos preocupar com o \emph{espaço} que esse texto
11 ocupa em nosso editor. Precisamos apenas olhar para a numeração das linhas no
12 arquivo original em LaTeX e verificar que cada parágrafo está em uma única linha
13
14
15
16 E o que acontece se deixarmos várias linhas em brancos? Vamos testar?
17 \end{document}

```

Line 17, Column 15      Tab Size: 4      LaTeX

Imagen do código do arquivo “lutex04.tex”

Depois de gerarmos o arquivo, teremos o seguinte resultado:

```

1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5 Nessa primeira linha, colocaremos várias frases. No entanto, como não iremos não
6 iremos assinalar ao LaTeX nenhuma quebra de linha, todas elas ficarão no mesmo
7 parágrafo.
8
9 Para assinalarmos que estamos em um novo parágrafo, deixamos a linha anterior em
10 branco. Novamente: não precisamos nos preocupar com o \emph{espaço} que esse texto
11 ocupa em nosso editor. Precisamos apenas olhar para a numeração das linhas no
12 arquivo original em LaTeX e verificar que cada parágrafo está em uma única linha
13
14
15
16 E o que acontece se deixarmos várias linhas em brancos? Vamos testar?
17 \end{document}

```

Line 17, Column 15      Tab Size: 4      LaTeX

Imagen do código do arquivo “lutex04.tex”

Os primeiros parágrafos saíram exatamente de acordo com aquilo que já discutimos: todo o

conteúdo que está em uma mesma linha gera um parágrafo; se houver apenas uma quebra de linha entre duas frases, o LaTeX *também* as mantém no mesmo parágrafo; se houver uma linha em branco, ele inicia um novo parágrafo.

Espero que o leitor tenha notado que introduzi algo novo nesse exemplo: dessa vez, separei o parágrafo final por *várias* linhas em brancos. Quando nos voltamos para o resultado final, no entanto, vemos que nada de diferente aconteceu: esse parágrafo está separado exatamente pela mesma distância dos outros. Mais uma vez, isso nos remete a um princípio fundamental do LaTeX: não importa o espaço visual em nosso editor de texto, o que importa é a lógica adotada pelo LaTeX.

O exemplo do último parágrafo reforça o modo como o LaTeX lida com linhas em branco: não importa o número de linhas, quando o LaTeX encontra uma linha em branco ele simplesmente pressupõe que o parágrafo anterior acabou e que o texto seguinte deve ir para um novo parágrafo – independentemente de *quantas* linhas estavam em branco.

É importante entender a relação entre essa convenção e o nosso conhecido princípio da separação entre *estrutura lógica* e *aparência visual* do texto. A distância entre os parágrafos é uma questão que diz respeito à *aparência* do documento e não à sua *estrutura lógica*. A separação entre linhas serve apenas para organizar a estrutura do nosso documento, sem afetar em nada sua aparência.

Se nosso desejo for efetivamente alterar a aparência do documento, então precisamos utilizar comandos que digam para o LaTeX que este é nosso desejo expresso. Podemos fazer isso por meio de alterações globais (realizadas no preâmbulo) ou por alterações locais (comandos inseridos no corpo do documento). Em todo caso, isso *não* pode ser feito com a quebra de linha. A divisão entre linhas não tem função visual, mas apenas função lógica; de modo que as linhas em branco não afetam a *quantidade* de espaço, mas apenas assinalam que começou um novo parágrafo.

Nesse sentido, note-se que isso nos livra da preocupação em aumentar e diminuir os espaços para fazer o texto caber em um número específico de páginas. Essas soluções dos editores convencionais são, por natureza, sempre provisórias e fatalmente terminam embaraçadas por mudanças posteriores na formatação. Ao decidirmos questões relativas à distribuição de espaço por critérios lógicos, nosso texto fica muito mais estável e controlável.

Em suma, vimos que o LaTeX possui um modo bastante específico de lidar com linhas. Ao redigir um documento, devemos utilizar as separações intencionais entre linhas de modo consciente para demarcar a estrutura lógica do nosso texto. Podemos ignorar as quebras meramente visuais de linhas, mas precisamos prestar atenção à numeração das linhas. Elas possuem uma função muito importante: elas delimitam como os elementos textuais devem estar relacionados, isto é, elas delimitam quando os parágrafos começam e terminam.

Para fazer isso, precisamos ter em mente o comportamento convencional do LaTeX diante das quebras de linha. Podemos resumir o que vimos do seguinte modo: 1. Todo o texto presente em um única linha irá ser alocado em um mesmo parágrafo. 2. Se houver uma única quebra de linha, o LaTeX irá entender que você está apenas organizando o texto para si mesmo e *não* começando um novo parágrafo. 3. Se você deixar uma linha em branco, o LaTeX irá entender que você começou um parágrafo inteiramente novo. 4. Se você deixar *várias* linhas em branco, o LaTeX *também* irá entender que você simplesmente quis começar um novo parágrafo e, portanto, irá deixar o mesmo intervalo de espaço que deixará se houvesse apenas uma linha em branco.

## A importância das linhas na solução de erros

Embora isso já tenha sido citado, é importante reforçar a importância das linhas para resolver problemas que surgem no momento de *gerar* os arquivos. Por exemplo, no seguinte exemplo, pegamos o arquivo “lutex01.tex” e introduzimos o seguinte erro na linha 7: vamos trocar o comando “\emph” (que deixa o texto selecionado em itálico) pelo comando “\emhp” (um erro de digitação comum que gera um comando inexiste. Observe a mensagem de erro do LaTeX:

```
./lutex01.tex:6: Undefined control sequence. [O \emhp]
```

Embora ocasionalmente a mensagem seja redigida de modo estranho, com um pouco de atenção podemos descobrir o aconteceu. A mensagem “Undefined control sequence” indica que tentamos utilizar um comando que não foi definido. Imediatamente na sequência, o LaTeX imprime a passagem do nosso texto onde o comando problemático foi utilizado. Além disso, se voltarmos ao arquivo fonte, veremos que esse erro ocorreu na sétima linha do arquivo “lutex01.tex”. O início da mensagem de erro aponta justamente para esse trecho do arquivo: “lutex01.tex:6”. Ou seja, esse número *depois* do nome do arquivo indica justamente a linha onde o problema ocorreu.

Embora seja sempre preciso um pouco de esforço para decifrar essas mensagens (que podem ser mais ou menos claras dependendo do erro), se ficarmos atento aos números que aparecem, em geral serem remetido à linha exata onde o erro aconteceu. De posse dessa informação, fica incrivelmente mais fácil resolver o problema.

Note-se que há alguns erros que impedem o LaTeX em apontar para linha exata onde o problema ocorreu. Às vezes, é possível que o erro remeta o LaTeX à linha imediatamente posterior ou anterior; em ambos os casos, no entanto, a informação se revela útil. Note-se também que nem todo erro pode ser remetido a uma linha específica. No entanto, via de regra, o LaTeX tentará – nem sempre com sucesso – identificar a linha que gerou o problema.

## Como o LaTeX interpreta os espaços

Em relação aos espaços entre as palavras, o LaTeX faz algo análogo ao que acontece com o espaço entre linhas: ele os trata como uma *separação lógica* entre as palavras e não como indicativo da *quantidade* de espaço. Novamente, a quantidade de espaço é definida automaticamente a partir de um cálculo relativo ao melhor modo de distribuir as palavras dentro das características gerais do documento definidas no preâmbulo. Colocar mais ou menos espaços entre as palavras simplesmente não afeta a *apresentação* final do documento.

Com exemplo, vejamos o código do documento “lutex05.tex”:

```
\documentclass{article} \usepackage{xltxtra}  
\begin{document}
```

```
Uma frase com vários espaços diferentes entre as palavras.
```

```
Uma frase com apenas um espaço entre as palavras
```

```
\end{document}
```

Como poder ver no exemplo acima temos bastante espaço desnecessários entre as palavras na primeira linha do texto. Enquanto isso, na segunda linha de texto, há apenas um único espaço. No entanto, isso não faz diferença no resultado final. Como podemos ver na imagem do documento gerado por esse exemplo (lutex05.pdf), o LaTeX *ignora* os espaços adicionais entre as palavras:



Imagen do arquivo “lutex05.pdf”

Em suma, a quantidade de espaços no código-fonte do arquivo em LaTeX *não* afeta a distância entre as palavras no arquivo final em formato PDF. O LaTeX considera que, independentemente no número de espaços entre duas palavras, o autor simplesmente assinalou os limites lógicos entre uma palavra e outra. A distância visual entre as palavras – assim como em relação às linhas – também é calculada automaticamente pelo LaTeX a partir das características gerais do documento definidas no preâmbulo.

## O problema das aspas

O leitor certamente já percebeu que o LaTeX é uma ferramenta desenvolvida e utilizadas por pessoas que possuem uma preocupação especial com tipografia. Por causa disso, essas pessoas dedicam uma quantidade enorme de atenção à problemas que parecem banais para a maioria

das pessoas. É exatamente isso que ocorre com as aspas. Normalmente, pensamos nelas como sinais gráficos simples que utilizamos para indicar um termo especial ou iniciar uma citação.

No entanto, existem diferentes modos tipográficos em verter os sinais das aspas de acordo com a língua, o editor de texto e a fonte utilizada. Por isso, ao redigir um documento em LaTeX, precisamos tratar as aspas também como *elementos lógicos* do texto: as aspas que você vê em seu documento fonte não são as aspas que aparecerão no documento final; o LaTeX irá “ler” seu documento em busca dessas “aspas-códigos” e irá imprimí-las na forma ideal no documento final.

Felizmente, é bastante simples utilizar os códigos para aspas no documento fonte. O LaTeX trata de modo distinto os comando para “abrir aspas” e “fechar aspas” e, por isso, ele possui um símbolo distinto para cada função. O sinal para abrir aspas é parecido com o *acento grave*, porém, ele deve ficar “solto” no documento e não associado a nenhuma vogal. Do mesmo modo, o sinal para fechar aspas é análogo ao acento agudo, devendo também ficar solto no documento. Para utilizar aspas duplas, basta repetir os comandos.

Como cada teclado, dependendo das configurações de localização, possuem um modo distinto de criar esses sinais, não iremos explicar ao leitor como fazê-lo. No entanto, olhando para seu teclado, ele deve encontrar esses sinais gráficos. Provavelmente, terá que apertar a tecla correspondente e *depois* apertar novamente a tecla de espaço para deixá-los “soltos”, isto é, para que eles não sejam associados a nenhuma letra específica.

Para tornar isso mais claro, segue uma imagem desses sinais tais como eles aparecem em um editor. Olhando para eles, o leitor facilmente identificará seus equivalentes em seu teclado. Note-se que o LaTeX leva tão à sério a distinção entre abrir e fechar aspas que alguns editores chegam a deixar o trecho entre aspas em uma cor diferente, para você ter certeza que não esqueceu nada ao longo do documento.

Vamos ver como isso funciona na prática. Em primeiro lugar, vamos ver essa imagem do código do arquivo de exemplo “lutex06.tex”:

A screenshot of the Sublime Text editor showing a LaTeX document named 'lutex06.tex'. The code is as follows:

```
1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5
6 No \LaTeX, é ``muito'' importante lembrar de fechar as aspas.
7
8 \end{document}
```

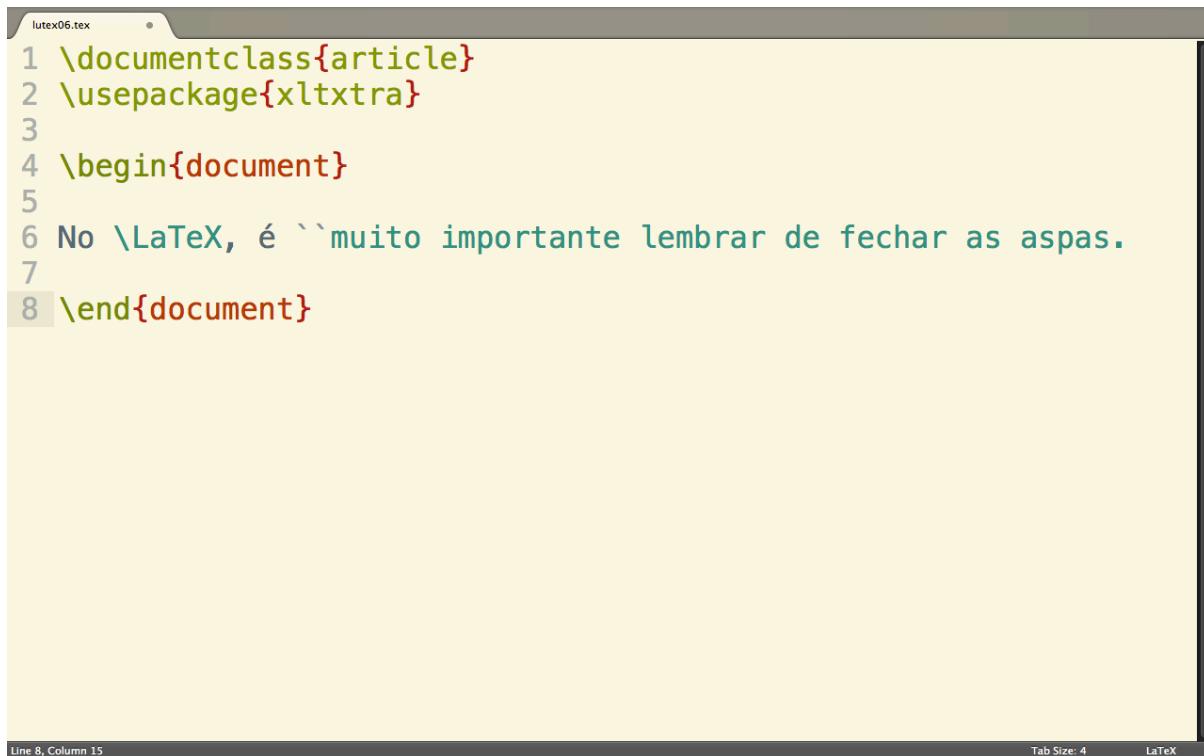
The line 'No \LaTeX, é ``muito'' importante lembrar de fechar as aspas.' is highlighted with a yellow background. The status bar at the bottom shows 'Line 8, Column 15' on the left and 'Tab Size: 4 LaTeX' on the right.

Imagen do código do arquivo “lutex06.tex”

Existem duas coisas novas para o leitor nessa passagem. A primeira delas é que, ao invés de simplesmente escrever LaTeX, eu coloquei o comando “\LaTeX”. Esse comando serve para colocar a expressão LaTeX em uma grafia especial – como veremos quando passarmos para a imagem do documento final.

Mas vamos voltar ao assunto deste seção: o leitor certamente percebeu que a palavra “muito” estava cercada de acentos diferentes. Estes são os sinais gráficos do LaTeX para assinalar a abertura e o fechamento das aspas. Como o Sublime Text sabe que o LaTeX leva esses dois elementos bastante à sério, ele deixou o conteúdo que estava entre aspas em uma cor diferente, para tornar mais fácil para o autor navegar pelo seu texto.

Note-se agora o que teria acontecido se tivéssemos esquecidos de fechar as aspas:



```
lutex06.tex
1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5
6 No \LaTeX, é ``muito importante lembrar de fechar as aspas.
7
8 \end{document}
```

Imagen do código *modificado* do arquivo “lutex06.tex”

Nesse exemplo, vemos que todo o resto do texto ficou em uma cor diferente. Essa mudança de cor serve para lembrar ao autor que algo estranho ocorreu: ele precisa voltar para o começo da mudança de cor e se lembrar em fechar suas aspas! Desse modo, um efeito colateral em utilizar o LaTeX é que se tornar muito mais difícil esquecer aspas flutuando livremente pelo texto.

Enfim, vamos agora ver o que acontece quando geramos o arquivo “lutex06.pdf” a partir da versão *correta* do arquivo “lutex06.tex” do exemplo anterior. Eis o resultado:

No **LATEX**, é “muito” importante lembrar de fechar as aspas.

Imagen do arquivo “lutex06.pdf”

Como podemos ver, os acentos agudos e graves que estavam “flutuando” ao redor do texto foram convertidos corretamente em sinais de abrir e fechar aspas.

É curioso também notar o que aconteceu com o nosso comando “\latexit”: ele gerou uma aparência bastante peculiar para o termo “LaTeX”. Esse comando tem sua raíz na história do LaTeX. Sua primeira versão foi desenvolvida por Donald Knuth, como um modo de simultaneamente criar uma identidade visual para o TeX e mostrar suas capacidades tipográficas. Como o LaTeX foi desenvolvido a partir do TeX, optou-se por criar um comando para fazer o nome “LaTeX” ficar parecido com a identidade visual do TeX original.

Mas voltemos à questão das aspas. É preciso se lembrar dos seguintes pontos. 1. O LaTeX utiliza um comando distinto para abrir e fechar aspas. 2. Para abrir aspas, utilize o comando equivalente ao sinal grave e aperte a tecla de espaço para deixá-lo solto no documento. 3. Para fechar aspas, utilize o comando equivalente ao sinal agudo e aperte a tecla de espaço para deixá-lo solto no documento. 4. Para utilizar aspas duplas, basta repetir o comando duas vezes, como vimos no exemplo acima.

## Divisão silábica no LaTeX

Já sabemos que o LaTeX organiza automaticamente nossas frases em parágrafos de acordo com os critérios de formatação definidos no preâmbulo. O LaTeX é inteligente o suficiente para fazer um cálculo e descobrir até que ponto pode aumentar o espaço entre as palavras sem torná-lo esquisito. Desse modo, ele evita o que frequentemente ocorre quando estamos lidando com editores convencionais: que, ao justificar o texto, as palavras fiquem espaçadas demais.

Para evitar isso, o LaTeX automaticamente calcula se as palavras estão apertadas ou folgadas demais e procura separá-las de acordo com as regras de separação silábica. Por isso, ao escrever um texto em LaTeX não precisamos nunca nos preocupar em realizar pessoalmente essa separação nem pensar em como o texto ficará justificado. É preferível simplesmente deixar o LaTeX realizar esse cálculo automaticamente e, depois, verificar se tudo foi realizado corretamente.

No entanto, ocasionalmente, as regras internas de separação silábica do LaTeX falham. Para evitar essas falhas, no entanto, há uma solução bastante simples: fazer o LaTeX carregar um dicionário para verificar a divisão silábica correta nas línguas utilizadas. Há também soluções manuais: você corrigir individualmente a divisão silábica de cada palavra ou mesmo *impedir* que o LaTeX separe uma palavra específica que você não quer ver divida. Em suma, como sempre no LaTeX, há várias soluções disponíveis. Nesta seção, veremos algumas delas.

Mas antes de entrar nas soluções, queria apenas observar que a eficiência do LaTeX em encontrar a divisão silábica mais adequada é realmente notável. Confesso que levei bastante tempo para encontrar um exemplo onde a divisão saiu errada. Em todo caso, eis o código do arquivo onde o LaTeX cometeu um erro:

```
\documentclass{article}
\usepackage{xltextra}
\begin{document}

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

\end{document}
```

Vamos agora ver como saiu o resultado final:

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

#### Imagen do arquivo “lutex07a.pdf”

Como podemos vemos no exemplo acima, o LaTeX acertou na maioria dos casos. No entanto, por algum motivo, ele não acertou a divisão da palavra “alterado”.

Como é bastante comum no LaTeX, há *várias* soluções possíveis para esse problema, cada uma mais adequada para cada caso. Uma solução possível é utilizar um *pacote* que avise ao LaTeX que seu texto está em português. Como vimos anteriormente, os *pacotes* são programas auxiliares que ampliam as funcionalidades do LaTeX. Eles devem ser adicionados ao preâmbulo do nosso documento e afetam as características globais do nosso texto.

Existem inúmeros pacotes que fazem o LaTeX carregar uma espécie de dicionário que o ensina a lidar corretamente com diferentes línguas. Para essa finalidade, recomendamos um pacote em particular: o *polyglossia*. A vantagem desse pacote é que ele é bastante popular (e, por isso, possui muita discussão sobre como utilizá-lo na internet) e ele é bastante potente. Ele tanto serve para ensinar o LaTeX a lidar com nosso texto em português como pode ser utilizado em textos com várias línguas – inclusive, línguas “mortas” como o grego antigo. Como os pesquisadores acadêmicos frequentemente precisam lidar com citações nas mais diversas línguas, achamos que vale a pena utilizar o *polyglossia* desde o início.

Para utilizar o pacote *polyglossia* não basta simplesmente chamá-lo utilizando o comando “\usepackage”; é preciso também utilizar alguns comandos próprios para informá-lo *qual* a língua especificamente utilizada no documento. (Para saber os comandos específicos de cada pacote, basta consultar o manual de utilização que geralmente acompanha a instalação de cada pacote).

Vamos pegar exatamente o mesmo arquivo do exemplo anterior e – sem fazer qualquer modificação no corpo do documento – vamos modificar apenas o preâmbulo do documento anterior adicionando duas novas linhas:

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{polyglossia} % Essa linha diz para o LaTeX utilizar o pacote “polyglossia”.
\setmainlanguage[brazil] % Essa linha é um comando que avisa ao pacote polyglossia para
utilizar o português brasileiro (para ver o comando de outras línguas, verifique a documentação
do polyglossia).
```

Depois de salvarmos o nosso novo arquivo (em nossa pasta de exemplos está como “lutex07b.tex”) e rodá-lo pelo LaTeX, teremos o seguinte resultado:

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

Imagen do arquivo “lutex07b.pdf”

Como podemos ver na imagem acima, nosso problema foi resolvido: o LaTeX continua utilizando a divisão silábica para organizar melhor o texto, mas não está mais comentando erros em relação à palavra “alterado”.

Caso por algum motivo você não possa utilizar o pacote *polyglossia* (alguns pacotes são incompatíveis entre si), você pode utilizar outros pacotes que “ensinem” ao LaTeX a ler seu texto em português. Sempre há soluções novas sendo propostas, de modo que não é muito difícil encontrá-las na internet.

Existe também uma solução específica para quem deseja utilizar as normas da ABNT: utilizar as *classes* de documentos preparadas pela equipe do ABNTex2. Como vimos anteriormente, a *classe* de um documento determina o *tipo geral* do texto que estamos escrevendo, afetando todas

as suas características textuais. Embora um documento possa ter vários pacotes, ele pode ter apenas uma classe.

A equipe abnTeX2 criou uma série de classes para cobrir as necessidades dos pesquisadores brasileiros. Essas classes carregam automaticamente uma série de definições relativas ao tipo de papel, à fonte, às margens do texto e várias outras características do seu documento. Por isso, ela deve ser utilizada somente por quem realmente tem interesse em colocar *todo* o seu documento dentro das regras da ABNT. Se você quer apenas corrigir seus problemas de divisão silábica, um pacote como o *polyglossia* oferece uma solução mais pontual.

No entanto, se você realmente vai seguir as normas da ABNT, então você não precisa do *polyglossia*: as classes do abnTeX2 já informam ao LaTeX que seu texto está escrito em português. Portanto, você possui dois tipos de soluções bem diferentes. Se quiser mudar *todo* o documento, utilize uma classe nova; se quiser modificar *apenas um aspecto* do seu documento, utilize um pacote específico.

Vamos ver agora como ficará o preâmbulo do arquivo “lutex07c.tex” com a inclusão da classe abnTeX2 (o corpo do documento será o mesmo):

```
\documentclass{abntex2} % Alteramos a opção “article” pela opção “abntex2”.  
\usepackage{xltextra} % Continuamos utilizando nosso pacote para lidar com as fontes.
```

Note-se também que, ao contrário, do *polyglossia*, o abnTeX2 é uma *classe* e não um *pacote*. Por isso, não utilizamos o comando “\usepackage”. Ao invés disso, trocamos a classe anterior (“article”) por uma nova classe (“abntex2”) no documento “\documentclass”. Note-se que esse exemplo é bem mais simples que o anterior. A classe “abntex2” já é suficiente para dizer ao LaTeX que nosso documento está em português, então podemos retirar as linhas relativas ao *polyglossia*. O código completo está na seção de anexos como “lutex07c.tex”.

Novamente, vamos ver o resultado final, o nosso arquivo “lutex07c.pdf”:

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

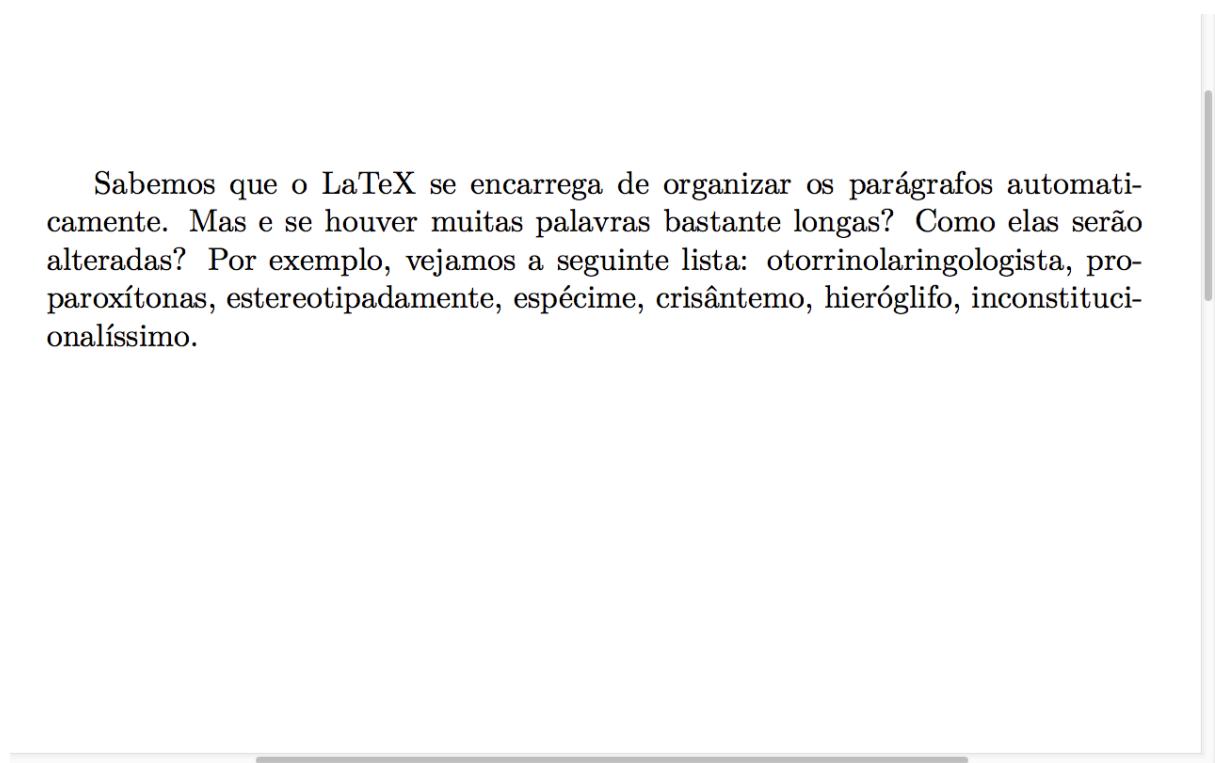


Imagen do arquivo “lutex07c.pdf”

Caso o leitor compare os dois exemplos anteriores, verá uma diferença importante entre as duas soluções. Em ambos os casos, tanto a classe “abntex2” quanto o pacote “polyglossia” resolveram o problema da divisão silábica. No entanto, a aparência final do documento está significativamente diferente entre os dois casos. As margens do documento “lutex07c.pdf” estão consideravelmente mais afastadas do que as margens do documento “lutex07b.pdf”; além disso, a fonte e o recuo do parágrafo estão diferente em ambos os casos.

Essas discrepâncias mostram bem a diferença entre utilizar uma *classe* e um *pacote* para resolver o mesmo problema. O *pacote* apenas adiciona mais algumas funcionalidades ao seu documento, enquanto a *classe* modifica todas as características globais de todo o documento. Nesse caso, a nova aparência do exemplo acima se deve ao fato de que a classe “abntex2” está colocando todo o documento dentro das normas da ABNT, *além* de ensinar ao LaTeX a utilizar as regras de divisão silábica do português. O pacote “polyglossia”, enquanto isto, está *apenas* ensinando o LaTeX as regras de divisão silábica, sem afetar as configurações globais do documento (que, no caso do exemplo anterior, estão seguindo as configurações padrões da classe “article”).

Em suma: se quiser corrigir a divisão silábica *e* utilizar as regras da ABNT, utilize as classes do abnTeX2 (note-se que é preciso instalá-las separadamente, dependendo da sua distribuição do LaTeX). Se quiser apenas corrigir as regras de divisão silábica enquanto utiliza outras configurações globais de aparência para seu texto, apenas adicione um pacote de língua e informe ao LaTeX que você está escrevendo em português.

Porém, há ainda inúmeras outras soluções possíveis para o mesmo problema – lembre-se de um dos lemas do LaTeX: *sempre há um novo comando*. Se você não quiser soluções globais e automáticas, existem vários comandos para interferir manualmente na divisão silábica de *cada* palavra.

Essa solução pode ser mais interessante nos seguintes casos: talvez você não queira carregar outro pacote ou talvez a palavra em questão não tenha na biblioteca que você está utilizando. Enfim, independentemente do motivo, você pode utilizar o seguinte comando para dizer ao LaTeX como lidar com a divisão silábica de cada palavra: `\hyphenation{lista de palavras}`.

Esse comando pode ser utilizado de dois modos: você pode utilizá-lo para ensinar a divisão silábica correta ou para *impedir* o LaTeX em dividir uma palavra que você quer manter unida. Vamos mostrar como utilizá-lo nos dois casos.

Primeiramente, vamos voltar ao nosso primeiro exemplo, onde o erro de divisão silábica ocorreu, e inserir o seguinte código em nosso preâmbulo:

```
\hyphenation{al-te-ra-das}
```

Esse comando irá ensinar ao LaTeX uma nova divisão silábica para a palavra “alteradas”. Tente rodar esse comando e veja como, dessa vez, o LaTeX irá dividir a palavra segundo essa divisão. Vamos salvar o novo arquivo como “lutex07d.tex” com o seguinte código fonte:

```
\documentclass{article}  
\usepackage{xltextra}  
\hyphenation{al-te-ra-das}  
\begin{document}
```

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

```
\end{document}
```

Quando nos voltarmos ao resultado do arquivo, veremos que a divisão silábica está corrigida (mas note-se que essa solução resolve *apenas* o problema da palavra “alteradas”):

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

#### Imagen do arquivo “lutex07d.pdf”

Outra alternativa é *bloquear* a divisão silábica de certas palavras. Para isso, basta utilizar o mesmo comando, mas dessa vez sem indicar *onde* fazer a divisão. Se fizermos isso, o LaTeX vai entender que ele não deve dividir as palavras dentre desse comando. Como exemplo, vamos ver o código-fonte do arquivo “lutex07e.tex”:

```
\documentclass{article}
\usepackage{xltextra}
\hyphenation{alteradas automaticamente}
\begin{document}

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

\end{document}
```

Como podemos ver na seguinte imagem do arquivo “lutex07e.pdf”, o código adicional impediu o LaTeX em dividir as duas palavras que estavam dentro do comando. Por isso, no documento final, veremos que a distância entre as palavras será adaptadas para que elas permaneçam inteiramente em uma única linha.

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

Imagen do arquivo “lutex07e.pdf”

Em suma, apresentamos três soluções possíveis para o problema da divisão silábica: 1. Você pode adicionar um pacote de língua (como o pacote *polyglossia*) e ensinar ao LaTeX a utilizar as regras de divisão silábica da língua indicada no comando. 2. Você pode utilizar uma classe que altere todo o documento para uma língua específica (como a classe *abnTeX2*), além de modificar outras características globais do documento. 3. Você pode utilizar um comando específico (`\hyphenation{lista de palavras}`) para ensinar ao LaTeX como dividir as palavras ou para *impedí-lo* de dividir as mesmas palavras.

Como sempre, em se tratando do LaTeX, há várias outras soluções possíveis. Caso o leitor queira algo mais apropriado para seu caso, basta buscar por “*hyphenation*” na internet que encontrará inúmeras soluções.

## Os sinais gráficos problemáticos

Existem mais alguns cuidados que devemos ter ao redigir um texto em LaTeX: alguns sinais gráficos não podem ser transcritos diretamente por terem um significado especial para o LaTeX. Já vimos que os documentos em LaTeX são uma combinação entre texto e códigos. Isso cria um problema: como o LaTeX sabe que deve considerar a próxima letra como parte do texto ou como parte do código?

Para fazer essa separação, o LaTeX possui uma série de convenções que guiam seu comportamento. Para isso, ele julga algumas letras como códigos especiais e não as imprime enquanto texto. Nesse sentido, quando ele encontra uma barra invertida () ele pensa: “Não devo imprimir essa barra, pois ela significa que o que se segue é um código”. Do mesmo modo, quando ele

encontra o sinal de porcentagem (%) ele pensa: “Não devo imprimir o que está no restante da linha, pois isso significa que o resto da linha é apenas um comentário para o próprio autor”.

Além desses dois caracteres especiais que já conhecemos, existem vários outros que o LaTeX *não* irá interpretar como texto comum. Alguns exemplos são os seguintes caracteres que *não* irão aparecer enquanto texto no LaTeX:

```
# $ % ^ & _ { } ~
```

Entretanto, isso não significa que é impossível utilizar esses sinais no seu texto. No entanto, para fazer isso, é preciso ter um pouco mais de trabalho: é necessário “avistar” ao LaTeX que, nesse caso específico, estamos realmente querendo utilizar esses sinais enquanto parte do texto e não como parte do código.

Para fazer isso, a regra geral é utilizar a própria barra invertida *seguida* do sinal especial. Por exemplo, se escrevermos em nosso texto o código “#” – sem as aspas, obviamente – o resultado no documento final será simplesmente o sinal “#”. Infelizmente essa regra possui um exceção: o uso de duas barras invertidas *não* mostra uma única barra, mas cria uma quebra de linha. Para fazer com que a barra invertida apareça em seu texto final, é preciso utilizar o comando “\textbackslash”.

Evidentemente, não é muito conveniente decorar todos esses casos especiais. É preferível simplesmente se lembrar da regra geral e pesquisar como resolver problemas específicos quando surgirem. Naturalmente, terminamos por decorar as soluções que utilizamos com maior frequência.

Lembrando as regras principais: 1. Se o LaTeX considera algum caractere especial, ele não irá aparecer como parte do texto. 2. Se quiser utilizar alguns desses caracteres como texto, coloque a barra invertida antes dele. (Acontece bastante de precisarmos utilizar o comando “\&” para introduzirmos o caractere “&” na bibliografia). 3. Para publicar a barra invertida em nosso texto, o comando “\” *não* funciona. Para isso, é preciso utilizar o comando “\textbackslash”.

## Notas finais

Nesse capítulo, nosso objetivo foi explicar o modo como o LaTeX interpreta o texto. Vimos que o fato dos textos em LaTeX serem uma combinação entre *códigos* e *texto* implica em complicações específicas, mas também em facilidades adicionais. Devemos nos lembrar que os espaços entre as palavras e os espaços entre linhas também são vistos pelo LaTeX enquanto parte dos códigos a serem analisados; sendo, por isso, interpretados logicamente e não em termos de espaço quantitativo.

As linhas do LaTeX são divididas em termos lógicos e não visuais. Por isso, é interessante utilizar um editor de texto que enumere as linhas utilizadas. O LaTeX irá automaticamente se encarregar de colocar todo o texto que estiver na mesma linha (ou separado por apenas uma linha) em um

único parágrafo. Para começar um novo parágrafo, é preciso deixar uma linha em branco entre cada porção de texto.

A quantidade de espaço entre parágrafos, entre palavras e no recuo do parágrafo é sempre decidida logicamente, a partir dos critérios gerais estabelecidos no preâmbulo. Enquanto estamos escrevendo, não devemos pensar na aparência final do texto, mas apenas nos critérios lógicos de organização interna. A aparência final é definida a partir da classe do documento e dos pacotes utilizados no preâmbulo. Os pacotes servem para introduzir funcionalidades específicas e a classe determina as características globais do documento.

Ocasionalmente, esse cálculo automático do LaTeX pode produzir resultados indesejados. Como ele tenta evitar que as palavras fiquem apertadas ou folgadas demais em um parágrafo, ele pode terminar separando a palavra por meio de uma regra incorreta de divisão silábica. Para evitar que isso ocorra, o modo mais efetivo é utilizar uma classe ou pacote que indique ao LaTeX que você está escrevendo em português e que ele deve utilizar as regras específicas da língua. Fora isso, é possível também corrigir cada palavra individualmente por um comando que foi explicado ao longo do capítulo.

Também vimos que alguns caracteres precisam de cuidados especiais. Em relação às aspas, o LaTeX precisa ser avisado quando nós queremos abrir aspas e quando queremos fechar aspas. Outros caracteres especiais não podem ser publicados diretamente: é preciso colocar a barra invertida antes deles para serem transcritos no documento final (e a própria barra invertida só será publicada por um comando específico para ela).

Enquanto neste capítulo tentamos explicar como o LaTeX lida com os textos, no próximo capítulo iremos nos voltar especificamente para discutir os comandos do LaTeX. Embora haja uma quantidade quase infinita de comandos, tentaremos apresentar aqueles que são mais úteis para o público-alvo deste livro.

# Capítulo 5: Os comandos do LaTeX

## Introdução

Ao longo desse livro, já fomos apresentados a alguns dos comandos do LaTeX. No entanto, optei por não fazer nenhuma apresentação exaustiva desses comandos por não considerar que essa é uma estratégia eficiente de aprendizagem. Além disso, simplesmente não é preciso decorar a maior parte dos comandos, pois eles são facilmente acessíveis na internet. Por fim, a grande maioria dos comandos é utilizada apenas uma vez para resolver problemas específicos (e não precisam ser decorados), enquanto aqueles poucos que utilizamos com frequência terminam por ser decorados automaticamente (sem falar que podemos adicioná-los como teclas de atalho em nosso editor).

Por todos esses motivos, não tentaremos fazer uma lista exaustiva dos comandos. Queremos apenas explicar aqueles que são especialmente relevantes e, no processo, transmitir os fundamentos do assunto que ajudarão o leitor a encontrar os comandos adicionais em guias mais específicos. Note-se que há comandos nativos ao LaTeX e há comandos que são adicionados por pacotes e classes especiais. Portanto, é importante procurar informações específicas nos guias relevantes ao tópico em questão.

Em parte, essa multiplicidade de comandos se deve justamente à flexibilidade do LaTeX. Como ele é inspirado na filosofia do *código aberto*, qualquer pessoa com conhecimento suficiente pode criar seus próprios “pacotes” e ampliar as funcionalidades do LaTeX. Por isso, existem milhares de pessoas no mundo inteiro continuamente desenvolvendo novos sistemas auxiliares para ampliar aquilo que o LaTeX é capaz de fazer. O efeito colateral disso é que existem diversos modos de fazer a mesma coisa e, portanto, comandos diferentes para os mesmos objetivos.

Justamente por isso não faz sentido aprender *todos* os comandos: cada pacote trará seu próprio conjunto de comandos e sempre haverá um pacote novo.

Para lidar com essa dispersão, os usuários do LaTeX desenvolveram duas estratégias: 1. Ele produzem uma “documentação” para cada novo pacote e classe, que consiste em um guia que apresenta uma descrição exaustiva sobre tudo relacionado ao pacote ou à classe em questão. 2. Eles se reunem em comunidades virtuais para resolverem problemas comuns.

Por isso, quando o leitor estiver diante de um novo problema, deve sempre tentar seguir esses passos em sequência: (i) procure um pacote que resolva seu problema; (ii) leia a documentação anexada para aprender a utilizá-lo; (iii) busque os fóruns de discussão para ler as discussões anteriores; (iv) peça ajuda fazendo referência aos passos realizados anteriormente – isso aumentará a sua chance de receber indicações relevantes.

Ao identificar erros em seu código, é sempre importante criar um “exemplo mínimo” do código problemático. Isso significa criar um novo arquivo onde você tenta reproduzir o erro isoladamente. Ao fazer isso, você tanto exclui a hipótese de que há um problema de compatibilidade entre pacotes e classes como fica mais fácil mostrar o arquivo para outros usuários do LaTeX para pedir ajuda.

## Características gerais dos comandos em LaTeX

Embora haja uma ampla diversidades de códigos em LaTeX, eles tendem a seguir algumas características gerais que o leitor já viu ao longo desse livro. Vamos revisá-las rapidamente antes de progredir para questões novas.

Como vimos nos capítulos anteriores, os códigos em LaTeX sempre começam com uma barra invertida () seguida pelo nome do código e por diferentes variáveis entre colchetes ([] e chaves ({}). Geralmente, as chaves trazem a principal variável, de modo que nem sempre o comando faz uso da opção entre colchetes. Em alguns casos, vemos que o código inicia um “comportamento” do LaTeX que precisa ser encerrado com outro código. Por exemplo, o comando “`\begin{document}`” precisa ser complementado com o comando “`\end{document}`”; são os códigos que criam um “ambiente”, dentro do qual, o LaTeX se comporta de acordo com um padrão distinto.

Por fim, o leitor precisa saber que os códigos em LaTeX não são necessariamente universais. Alguns códigos funcionam exclusivamente para documentos compilados com o XeLaTeX (a opção recomendada por esse livro); alguns códigos funcionam apenas na presença de uma classe específica; outros códigos só funcionam se determinado pacote estiver ativado. Além disso, existem códigos que devem ser utilizados *exclusivamente* no preâmbulo do documento, assim como códigos que devem aparecer *apenas* no corpo do texto.

Por isso, é interessante que o leitor não adicione muitos códigos novos de uma única vez em um novo documento. É preferível ir adicionando novos códigos aos poucos e compilando o documento com frequência para saber se tudo está sendo processado corretamente, o que torna mais fácil identificar e corrigir problemas.

Evidentemente, algumas pessoas preferem simplesmente baixar um modelo pronto e simplesmente começar a digitar, sem entender muito bem a razão de todos os códigos presentes do documento. Esse comportamento é perfeitamente aceitável. O LaTeX se baseia justamente na economia de esforço e na separação entre forma e conteúdo. Logo, não há algum problema algum em testar um preâmbulo novo preparado por outra pessoa com seu documento. Uma das vantagens do LaTeX é poder testar rapidamente diferentes formatos e estratégias de apresentação sem afetar o documento propriamente dito. No entanto, evidentemente o ideal seria que cada leitor compreendesse todos os comandos, para ficar mais fácil criar suas próprias adaptações – mas isso não é de modo algum necessário.

Em todo caso, no restante desse capítulo, tentarei apresentar alguns dos principais comandos do LaTeX para a produção de um trabalho acadêmico. Note-se que evitarei propositalmente a discussão referente à referências bibliográficas nesse capítulo, pois o tema possui complexidades que tornam preferível discuti-lo em um capítulo separado.

## Os comandos do preâmbulo: definindo a classe do documento

O leitor certamente já sabe que o *preâmbulo* define as *características globais* do documento. Como vimos anteriormente, o LaTeX presume que o preâmbulo começa na primeira linha

do documento e continua até o momento em que assinalamos que o texto propriamente dito começou. O leitor também já sabe que no preâmbulo nós definimos a *classe* do documento e os *pacotes* que irão adicionar funcionalidades ao nosso texto.

Embora o leitor já tenha visto o comando que estabelece a *classe* do documento, ainda não discutimos todas as suas possibilidades. Como podemos ver acima, esse comando admite dois tipos de variáveis. Como já discutimos, a regra geral se aplica neste caso: a variável principal sempre fica entre as chaves ({}), enquanto a variável secundária fica entre os colchetes ([]).

Embora a variável secundária apareça primeiro na linha de comando (o que pode gerar alguma confusão), ela é sempre dependente da variável principal. Em certo sentido, a variável principal “diz” ao LaTeX o que é possível esperar da variável secundária. No caso do comando “\documentclass”, dependendo da *classe* escolhida, o usuário terá diferentes opções que poderá acionar na outra variável.

É importante compreender como o LaTeX utiliza esse comando. Ele vai pegar o valor entre as chaves (o nome da classe) e procurar em seus arquivos por uma folha de estilo descrevendo uma série de características visuais do documento final desejado. Por isso, é necessário que o nome da classe utilizada em seu documento corresponda à uma classe que já esteja instalada em seu computador.

O LaTeX é bastante inteligente nesse busca: ele irá primeiramente procurar por esse arquivo no diretório onde está o seu arquivo original e, depois, irá procurar em sua própria biblioteca. Por isso, você tem a opção tanto em deixar o arquivo referente à classe desejada na pasta em que está trabalhando (o que é interessante apenas se for uma classe que você só pretende utilizar poucas vezes) ou deixá-la salva na pasta da biblioteca do LaTeX (o que é preferível se você for utilizá-la várias vezes).

Se você causar um erro de digitação ou utilizar uma classe que não está instalada em seu computador, o LaTeX lhe avisará que está faltando um arquivo “nome-da-classe.cls” em seu computador. Podemos ver isso se tentarmos rodar um arquivo com os seguinte código:

```
\documentclass{classe123}
\usepackage{xltxtra}
\begin{document}
Seu texto.
\end{document}
```

Se o leitor fizer esse documento, verá que nenhum arquivo “.pdf” será gerado. Antes disso, o LaTeX irá interromper seu processo e irá imprimir a seguinte mensagem de erro: “LaTeX Error: File ‘classe123.cls’ not found.”

Assim que você instala o LaTeX, ele já vez com várias classes instaladas automaticamente. No entanto, você talvez queira baixar novas classes. É possível fazer isso tanto manualmente como utilizando os sistemas de atualização automática que vêm em sua distribuição do LaTeX.

Em geral, você encontrará instruções sobre como instalar essas classes na página de quem as produziu.

Um caso que deverá interessar especialmente aos leitores desse livro são as classes desenvolvidas pelo grupo abnTeX2. Elas cobrem as diferentes necessidades acadêmicas (artigos, teses, relatórios) e colocam todo o seu documento nas regras da ABNT. Essas classes, no entanto, não vêm automaticamente instalados em todas as distribuições do LaTeX e precisam ser instaladas posteriormente<sup>[^7]</sup>. Apesar de serem extremamente úteis, não iremos aqui descrever em detalhe como instalá-las e utilizá-las. Esse tipo de projeto está sempre em constante evolução, de modo que qualquer descrição que fizermos sobre sua instalação e sobre seus comandos corre o risco de se tornar datado rapidamente. Por isso, simplesmente recomendamos fortemente os leitores para procurarem o site do grupo, onde terão acesso às informações mais recentes. Em todo caso, esperamos que as informações trazidas por esse livro sejam suficientes para ajudá-los a compreender como utilizar as classes produzidos pelo abnTeX2.

O site do grupo: <https://code.google.com/p/abntex2/><sup>5</sup>

Evidentemente, é possível também *criar* suas próprias classes. No entanto, uma das vantagens do LaTeX é justamente não precisar perder muito tempo com a aparência do documento, já que podemos simplesmente utilizar uma classe previamente criada por outra pessoa – com mais tempo e talento para isso. Portanto, só vale a pena criar uma classe inteiramente nova para os casos onde nenhuma classe existente pode – mesmo depois de algumas adaptações – resolver seu caso pessoal.

Já vimos uma das classes padrões do LaTeX: a classe *article*. Essa classe oferece um modelo mínimo de artigo científico com algumas características pré-carregadas. No entanto, é possível utilizar a variável secundária para alterar essas características: por meio dela é possível mudar o tamanho da fonte, o tipo de papel em que o artigo será impresso e mesmo avisar que a impressão será feita apenas de um lado ou será feita na frente e no verso do papel.

Como exemplo, recomendamos que o leitor pegue qualquer arquivo “.tex” utilizado anteriormente e veja o acontece quando se substitui a definição de classe alternadamente por cada uma dessas duas linhas de código:

```
\documentclass[11pt,twoside,a4paper]{article}
\documentclass[14pt,oneside]{article}
```

Note-se que, quando omitimos um valor (no segundo exemplo não há a opção pelo papel), o LaTeX utiliza a opção padrão da classe escolhida. Note-se também que essas opções estão disponíveis apenas especificamente para a classe *article*. Se você utilizar a classe abnTeX2, por exemplo, terá que utilizar as opções apropriadas para essa classe. Veja alguns exemplos possíveis:

---

<sup>5</sup><https://code.google.com/p/abntex2/>

```
\documentclass[12pt,twoside,a4paper,brazil]{abntex2}
\documentclass[10pt,oneside,a4paper,brazil]{abntex2article}
```

Nos dois casos, utilizamos opções parecidas com o exemplo anterior, mas a classe abnTeX2 possui uma função adicional (“*brazil*”) que indica que o texto que virá será em português brasileiro. Essa opção torna desnecessário a utilização de um pacote de línguas como o *polyglossia* para repassar as regras de divisão silábica ao LaTeX. Note-se ainda que é necessário possuir as classes do abnTeX2 instaladas para que elas possam ser utilizadas – o que não é necessariamente o caso de todas as distribuições do LaTeX.

Por fim, é importante lembrar que, como a escolha da classe afetará todo o restante do documento, ela afetará também os pacotes escolhidos e até mesmo o modo como o documento deverá ser redigido. Alguns pacotes e comandos podem se tornar desnecessários, redundantes e até mesmo incompatíveis com algumas classes. O pacote abnTeX2, por exemplo, possui suas próprias regras bibliográficas, de modo que não faz sentido escolhê-lo e depois utilizar um pacote adicional com comandos bibliográficos próprios. Portanto, é importante lembrar o princípio hierárquico de que a escolha da classe correta vem primeiro; o restante do documento – pacotes e comandos – se seguirá a partir disto.

## Os comandos do preâmbulo: adicionado pacotes ao seu documento

Por motivos evidentes, é possível atribuir apenas *uma* única classe ao seu documento. Afinal, a classe afeta todo o documento, dizendo para o LaTeX *o que* é que você está tentando escrever. Inversamente, você pode utilizar um número ilimitado de pacotes, pois cada pacote se limita a adicionar novas funções ao modo como o LaTeX pode ser texto. Portanto, exceto no caso de pacotes incompatíveis entre si (por exemplo, no caso de você utilizar dois pacotes que digam ao LaTeX para utilizar simultaneamente dois modos distintos de processar sua bibliografia), você pode utilizar quantos pacotes quiser.

Assim como em relação às classes, as distribuições de LaTeX já trazem vários pacotes pré-instalados. No entanto, como o número de pacote é absurdamente grande, sempre é possível encontrar novos pacotes que não vieram juntos com o LaTeX. Nesse caso, você também pode optar por instalá-los manualmente ou utilizar o administrador de pacote que tenha vindo com sua instalação do LaTeX.

Praticamente não há limite para o que um pacote pode fazer por seu documento. Basicamente, um pacote é um código produzido por algum usuário do LaTeX que amplia as funções do sistema em uma nova direção. A multiplicidade de pacotes é um dos aspectos mais interessantes de trabalhar com o LaTeX. Não importa qual seja a sua necessidade, há uma enorme probabilidade de que alguém tenha tido o mesmo problema e que já tenha desenvolvido um pacote especialmente para resolvê-lo.

Exatamente por isso, há uma diversidade muito maior de comandos em relação aos pacotes. Como vimos, o comando da classe fica inteiramente contido em uma única linha e é composta apenas de dois grupos de variáveis. O pacote começa com a mesma estrutura (`\usepackage[opções]{nome-do-pacote}`), mas pode admitir várias outras linhas de código com outras variáveis, dependendo das funcionalidades criadas pelo próprio pacote.

Vamos retomar, por exemplo, um dos pacotes que já utilizamos para definir a língua do nosso documento: o *polyglossia*. Como ele é um pacote que pode ser utilizado para documentos escritos em múltiplas línguas, ele também admite uma série de funções. Por isso, depois de utilizarmos o comando para “chamar” o pacote, nas linhas seguintes continuamos a adicionar características adicionais sobre como iremos utilizar o pacote.

Segue um exemplo de como podemos utilizar o pacote *polyglossia* para produzir um artigo bilíngue. O seguinte código deverá ser colocado em nosso preâmbulo, logo após a definição da classe. Adicionamos, ao final de cada linha, um comentário explicando cada comando:

```
\usepackage{polyglossia} % Aqui simplesmente avisamos ao LaTeX que vamos utilizar esse pacote.

\setmainlanguage{brazil} % Avisamos que a língua principal do nosso documento será o português do Brasil.

\setotherlanguage{english} % Adicionamos uma língua secundária.
```

Note-se que “*brazil*” e “*english*” são os termos oficiais presentes na documentação do pacote. É preciso usar o termo exato definido pela documentação associada ao pacote. Note-se que esse é apenas um exemplo mínimo da utilização desse pacote. É possível ainda adicionar várias outras funcionalidades, como fontes específicas para cada tipo de língua definida (também é possível definir várias línguas secundárias). Sempre vale a pena a documentação (o guia específico produzido pelos autores do pacote) que explica todas as funcionalidades de cada aplicação.

É possível também utilizar pacotes para alterar detalhes bastante específicos na aparência de um documento. Uma convenção textual nos países de língua inglesa é que o primeiro parágrafo de um capítulo, em certos tipos de documento, não possuem o recuo do início do parágrafo. Como não temos essa convenção em nossa língua, podemos “forçar” o LaTeX a abandoná-la. Para isso, basta utilizar o seguinte comando para inserir um pacote que irá alterar esse comportamento (como sempre, devemos adicionar os pacotes *dentro* do preâmbulo e *após* a definição da classe):

```
\usepackage{indentfirst} % Pacote que adiciona recuo ao primeiro parágrafo de cada seção.
```

Note-se que o pacote acima é bastante simples. Não é preciso definir nem uma variável secundária nem adicionar outros comandos, como fizemos o *polyglossia*. Nesse caso, o simplesmente ato de informar ao LaTeX que queremos utilizar esse pacote já é suficiente para alterar o documento do

modo desejado.

Note-se também que há pacotes que trazem o conteúdo de vários pacotes! Já vimos um exemplo deles: o *xltextra*. Esse pacote carrega, entre outras coisas, o pacote “*fontspec*”, que ensina ao LaTeX como lidar com a codificação dos acentos e caracteres especiais em nosso texto. No entanto, como já estamos utilizando o *xltextra*, não é necessário chamar especificamente o “*fontspec*” também, pois ele está sendo automaticamente ativado quando utilizamos o comando para dizer para o LaTeX utilizar o *xltextra*.

Aliás, é importante reforçar: alguns pacotes funcionam especificamente com compiladores específicos. Ao longo de todo o livro, estamos recomendando a utilização do XeLaTeX e, portanto, recomendamos que verifique se os pacotes utilizados são compatíveis com ele. Em outros guias, o leitor talvez encontra a recomendação de utilizar o pacote “*inputenc*” para resolver o problema da codificação dos acentos. Não há problema algum em utilizar esse pacote, mas ele *não é utilizado em conjunto com o XeLaTeX*. O “*fontspec*” também resolve o mesmo problema, mas é compatível com o sistema que recomendamos. Se tudo isso estiver muito complicado, vamos simplificar: lembre-se de compilar seus arquivos utilizando o XeLaTeX e utilize apenas o pacote “*xltextra*” para resolver seus problemas de codificação. Lembrando desses dois pontos, você não terá problemas com seus acentos e caracteres especiais.

Enfim, há uma quantidade praticamente ilimitada de pacotes que você pode utilizar em seus documentos. Sabendo disso, basta procurar um pacote específico para suas necessidades e estudar como utilizá-lo. Em alguns casos, basta “chamar” o pacote (como fizemos com o *xltextra*) *para resolver o problema; em outros casos, é preciso adicionar mais algumas linhas de comando para fazê-lo funcionar do modo desejado* (como fizemos com o *\_polyglossia*). Em cada caso, tudo dependerá especificamente das instruções relativas a cada pacote.

Por fim, é importante repetir que a escolha dos pacotes e das classes estão sempre estreitamente relacionadas tanto entre si e com o compilador escolhido. Desde o início do livro, estamos recomendando a utilização do XeLaTeX como sistema tipográfico para gerar os nossos arquivos. Isso nos levou a escolher os pacotes *polyglossia* e *xltextra*. Esses pacotes foram intrinsecamente preparados para operar com o XeLaTeX. Em outros guias, talvez você encontre a recomendação de utilizar o pacote *babel* para informar o LaTeX da língua utilizada e o pacote *inputec* para utilizar os caracteres latinos. Essa opção é igualmente possível *apenas* se for utilizada em conjunto, isto é, utilizando esses dois pacotes e utilizando o compilador padrão.

Preferimos, no entanto, utilizar esses dois outros pacotes adaptados especificamente ao XeLaTeX porque julgamos que eles possuem um suporte melhor para caracteres internacionais (veja nossa discussão sobre unicode e UTF-8 no início do livro). Como o público-alvo deste livro frequentemente lida com várias línguas estrangeiras – e muitas vezes com línguas exóticas, como grego clássico, hebreu e sânscrito – a utilização do XeLaTeX nos parece preferível desde o início do trabalho com o LaTeX. No entanto, é preciso indicar ao leitor que tudo isso são opções possíveis e que há várias alternativas aceitáveis. O importante é identificar a ferramenta adequada para cada trabalho e verificar se elas são competíveis entre si.

## Os comandos do preâmbulo: definindo o estilo do documento

Ainda no *preâmbulo*, há uma série de comandos que fazem pequenas alterações no estilo do documento. Mesmo tendo escolhido uma classe que atenda à maioria das nossas expectativas, podemos ainda querer alterar algum aspecto particular da aparência do documento.

Por exemplo, vimos que, ao escrever o texto, o espaço entre as linhas do nosso documento é interpretado pelo LaTeX apenas em termos lógicos, isto é, como indicativos do momento em que cada parágrafo começa e termina. Enquanto estamos escrevendo nosso texto – ou seja, dentro da seção indicada pelos comandos “`\begin{document}`” e “`\end{document}`” – simplesmente *não é o lugar* para alterar a aparência do documento. Nesse espaço, devemos apenas indicar a relação lógica entre os trechos do texto. Se quisermos realizar qualquer alteração global, devemos nos voltar para o preâmbulo.

Vimos nas seções anteriores que a classe indica as características globais do documento e que os pacotes adicionam funcionalidades específicas. Além desses comandos gerais, há comandos que alteram exclusivamente o *estilo* do documento. Novamente, é possível utilizar padrões previamente instalados no LaTeX ou baixar novos padrões.

É possível também utilizar comandos que alteram apenas um aspecto bastante específico do texto. Como exemplo, podemos citar os seguintes comandos que alteram a disposição dos parágrafos. Portanto, se estivermos insatisfeitos com as configurações padrões, podemos usar alguma variação do seguinte código, alterando o valor contido na seção dos centímetros:

```
\setlength{\parindent}{1.3cm} % Aumenta o tamanho do recuo do parágrafo para 1.3 centímetros.  
\setlength{\parskip}{0.2cm} % Aumenta o espaçamento entre parágrafos para 0.2 centímetros.
```

Evidentemente, esses códigos são estranhos demais para serem memorizados – e, novamente, isso felizmente não é necessário. Para esses códigos pontuais, podemos procurá-los quando necessário nos guias e na internet. Há códigos para realizar mudanças especificamente no cabeçalho, nas notas de rodapé, no sumário, na bibliografia, etc. Algumas vezes, esses códigos precisam de algum pacote para ser utilizado; em outros casos, são aceitos nativamente pelo LaTeX.

No entanto, uma regra prática que pode poupar bastante esforço é minimizar a utilização dessas soluções manuais para problemas específicos e identificar as folhas de estilo que resolvem o problema de modo global e automático. Novamente citando o exemplo do abnTeX2, temos um sistema completo que, a partir da definição da classe, já transmite todas as configurações necessárias para o documento inteiro. Entretanto, é verdade que nem sempre essas soluções completas estão disponíveis. Nesse caso, é importante lembrar que é possível também encontrar soluções que oferecem soluções específicas para problemas igualmente específicos.

## Os comandos do preâmbulo: Adicionando informações sobre o documento

Podemos também utilizar o LaTeX para salvar informações sobre o texto. Essas informações serão armazenadas pelo LaTeX e poderão serem eventualmente utilizadas de diferentes modos. Lembrando que o LaTeX tenta sempre separar *conteúdo* e *apresentação*, isso implica também em um modo distinto em lidar com as capas dos trabalhos, folhas de rosto, contra-capa, etc. Esses elementos textuais são bastante trabalhosos e precisam ser refeitos a cada vez. No entanto, eles sempre são frutos de uma combinação entre dois elementos bastante distintos: as informações que devem estar contidas nessas páginas e o modo como essas informações devem ser representadas.

Como sempre, o LaTeX tenta separar as categorias lógicas da sua apresentação final. Em alguns casos, ele tenta fazer isso automaticamente. Como veremos em relação ao sumário e à bibliografia, ele gera essas informações a partir das indicações lógicas presentes no próprio texto. No entanto, existem algumas informações que precisam ser informadas diretamente para que, depois, o LaTeX possa manipulá-las de diferentes modos.

Essas informações são, evidentemente, elementos opcionais; no entanto, é interessante nos acostumarmos a adicioná-las aos nossos documentos. É interessante observar também como diferentes classes e folhas de estilo modificam o modo como as mesmas informações são transmitidas. Como exemplo, recomendamos que o leitor adicione as seguintes informações ao *preâmbulo* seu documento:

```
\author{Nome do autor}  
\title{Título do documento}  
\date{\today}
```

A função de cada um desses três comandos é tão evidente que não requer explicações. O único detalhe a ser explicado está na terceira linha: note-se que, nesse caso, há um *comando dentro de outro comando*. O comando “\today” presente na terceira linha faz com que o LaTeX utilize como data do documento o dia em que o arquivo “.pdf” for gerado. A vantagem desse comando é que, a cada vez que você modificar seu texto e gerar uma nova versão final, a data será atualizada automaticamente. Evidentemente, haverá ocasiões onde é preferível deixar o documento sempre com a mesma data. Cabe, como sempre, ao leitor decidir o que opção é mais conveniente em sua situação pessoal.

Note-se ainda que o LaTeX não irá necessariamente utilizar essas informações registradas no preâmbulo. Essa utilização dependerá das características gerais do documento. Note-se que o leitor pode dizer ao LaTeX que deseja efetivamente utilizar essas informações por meio do comando “\maketitle” colocado *dentro* do corpo do documento. Esse comando irá gerar automaticamente um título pra seu documento, o que será feito a partir das folhas de estilo da classe escolhido (o que será bem diferente caso trata-se de um livro ou de um artigo acadêmico) e a partir das informações textuais registradas no preâmbulo; em outras palavras, diferentes classes

e pacotes possuem diferentes modos de fazer uso dessas informações.

Como exemplo, vamos utilizar o seguinte código e gerá-lo com duas classes diferentes:

```
\documentclass{article} % Teste modificar essa classe por "book" e note as diferenças.  
\usepackage{xltextra}  
\author{Nome do autor} % Sinta-se à vontade para modificar os valores dessas três linhas.  
\title{Título do documento}  
\date{\today}  
\begin{document}  
\maketitle % Esse comando diz para o LaTeX utilizar as informações registradas pelas últimas  
três linhas do preâmbulo.  
Seu texto.  
\end{document}
```

No primeiro caso, vamos utilizar a classe “article” e ver o resultado:

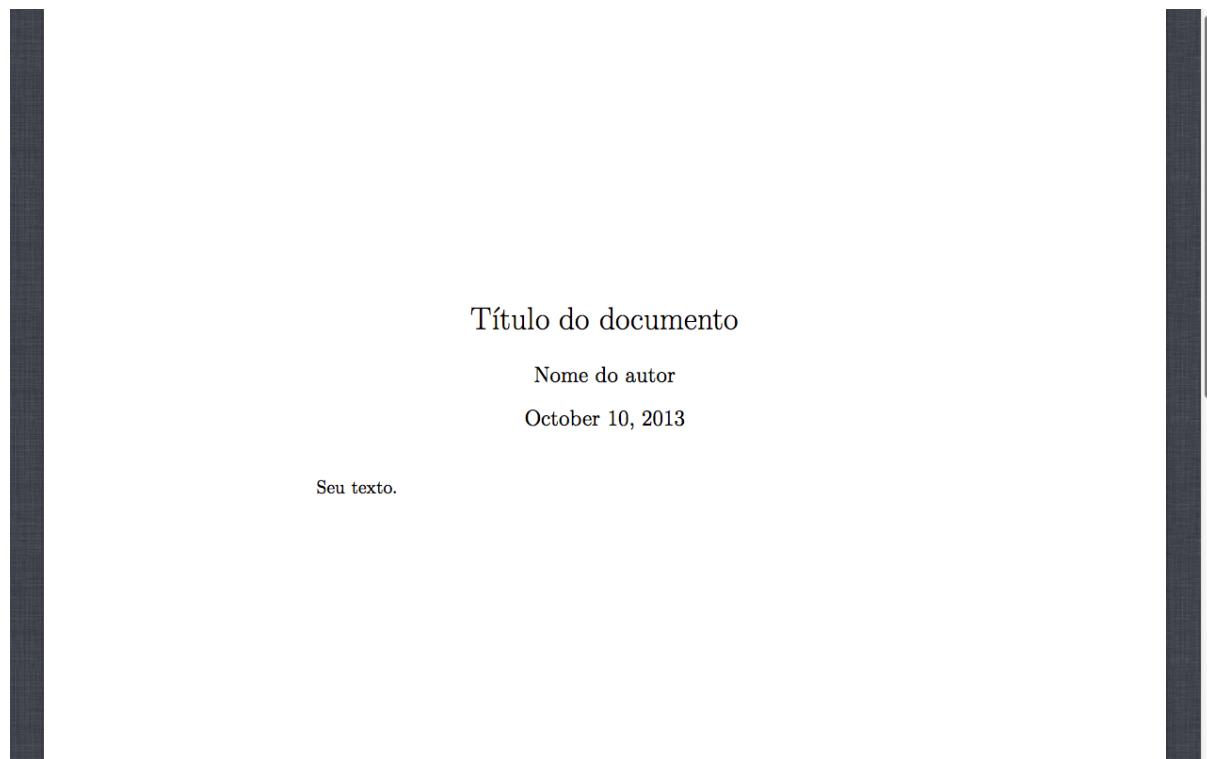


Imagen do arquivo gerado pelo código “lutex09a.tex” com a classe “article”

Vemos que o título foi gerado e o texto começa logo em seguida. Vamos ver o que acontece quando alternamos para a classe “book”:

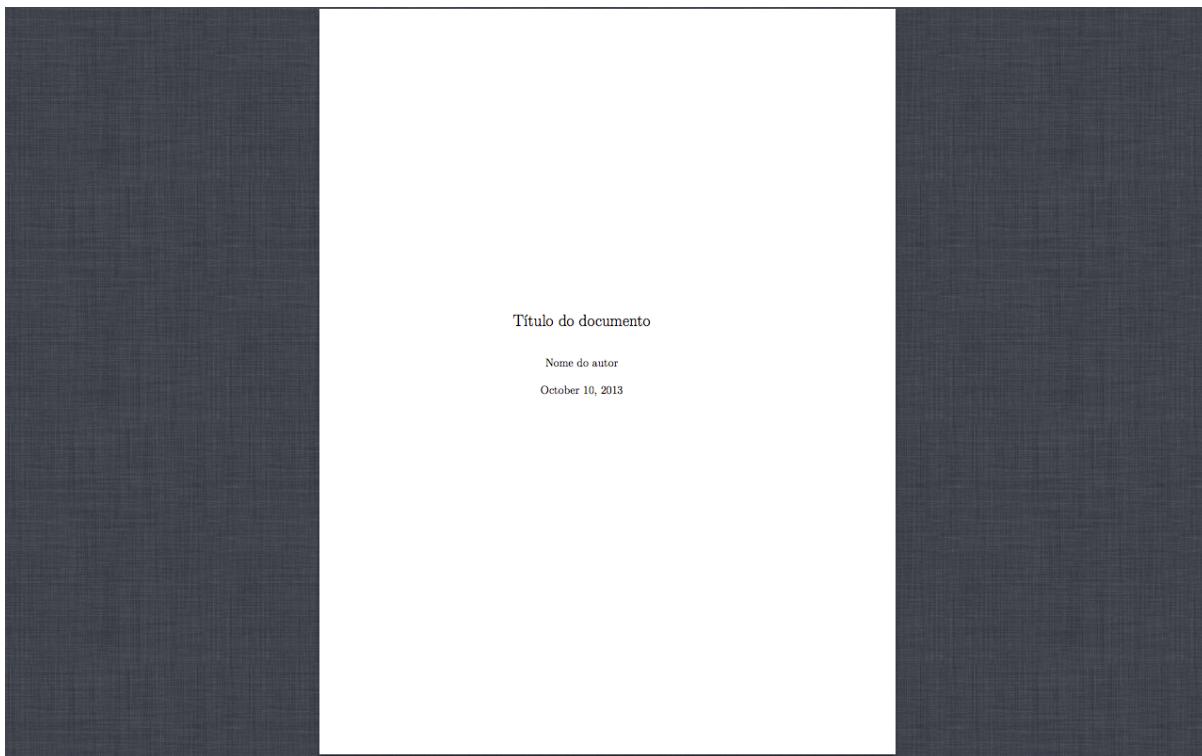


Imagen do arquivo gerado pelo código “lutex09b.tex” com a classe “book”

Nesse caso, o título do livro ocupou toda a primeira parte e o texto foi lançado para a segunda página do documento. Essa diferença é bastante razoável, já que os títulos dos livros normalmente ficam em uma página separada, enquanto os títulos dos artigos ficam logo acima do texto. Note-se que, em ambos os casos, o LaTeX gerou esses resultados automaticamente, considerando tanto a classe informada como as informações presentes no preâmbulo.

Esses exemplos são bastante simples, mas mostram o poder do LaTeX em gerar diferentes apresentações a partir do mesmo conteúdo. Existem classes mais elaboradas, como as classes da abnTeX2, que chegam ao ponto de gerar a capa, a contra-capa e mesmo a folha de rosto dos seus documentos. Note-se, no entanto, que para usar essas classes é preciso também seguir suas convenções e utilizar seus comandos específicos. Como sempre, não há soluções e comandos universais; é preciso verificar a documentação específica dos pacotes e classes utilizados.

Note-se ainda que até o modo como *registramos* essas informações (utilizando os comandos “\author” ou “\title”, etc.) e como a *utilizamos* (com o comando “\maketitle” *dentro* do documento) vai depender inteiramente da classe. Esses comandos estão em inglês porque estamos utilizando classes padrões do LaTeX que foram produzidas em inglês. Como o leitor poderá descobrir se instalar o sistema abnTeX2, ele poderá utilizar comandos em português para realizar as mesmas coisas.

## O conteúdo do documento: criando a estrutura do texto

Um dos efeitos colaterais mais interessante em escrever utilizando o LaTeX é que ele nos força a pensar em nosso texto de um modo mais organizado. O leitor, se já trabalhou em um

trabalho mais longo, sabe o quanto é tentador passar bastante tempo modificando a fonte de um novo subtítulo de uma seção, pensando se é melhor deixá-la em negrito ou sublinhá-la ou simplesmente apagar o subtítulo e continuar na seção anterior. Evidentemente, o LaTeX não irá organizar o texto por você: ainda é seu papel escolher o melhor modo de estruturar seu documento. O que o LaTeX fará será tornar essa organização mais evidente e tornar automática a definição da sua aparência.

O LaTeX nos força a pensar em termos estruturais justamente por causa da separação entre *conteúdo* e *aparência*. Em um editor convencional, nós criamos uma nova seção digitando o título da seção e atribuindo alguma grafia especial ao título. No caso do LaTeX, nós devemos utilizar um comando específico para indicar cada seção do texto. O comando básico para criar uma seção em LaTeX é “`\section{título da seção}`”. É possível criar seções subordinadas com os seguintes comandos análogos: “`\subsection{título}`” e “`\subsubsection{título}`”. Se for desejado, é possível criar uma quantidade quase infinita de subseções, mas a partir desse ponto é preciso utilizar novos comandos (“`\paragraph{...}`”, “`\ subparagraph{...}`”, etc.).

O modo como o LaTeX lida com a estrutura do texto depende inteiramente do *tipo* de documento em questão. Evidentemente, um livro e um tese exigem uma estrutura significativamente diferente de um artigo acadêmico. Para ambos os casos, no entanto, “`section`” denota a unidade básica da divisão estrutural do texto. No entanto, para projetos maiores, é possível utilizar categorias maiores: livros e teses admitem categorias que *englobam* as seções. Você pode reunir suas seções em capítulo (com o comando “`\chapter{título do capítulo}`”) e reunir vários capítulos em uma parte do livro (com o comando “`\part{título da parte}`”). No entanto, nem todas as classes utilizam essas categorias maiores: a classe “`article`” não utiliza essas divisões, já que artigos acadêmicos não costumam possuir capítulos e partes.

O LaTeX vai varrer todo o texto em busca dessas indicações lógicas para descobrir a estrutura fundamental do texto. Com base nessas informações, ele irá decidir uma série de questões relativas à apresentação do documento: ele irá numerar as seções e criar cabeçalhos, títulos, sumários, etc. Novamente, como sempre, todas as questões de estilo serão definidas pelo preâmbulo. Por isso, é perfeitamente possível separar a estrutura lógica do texto do modo específico como ele será apresentado. Enquanto estiver escrevendo, o autor deve se concentrar exclusivamente nessa estrutura e deixar que o LaTeX resolva o resto depois.

Vamos agora ver como isso funciona na prática. Logo abaixo, está o código fonte do arquivo “`lutex10a.tex`”, que foi dividido em várias seções e subseções. Como o tipo de arquivo utilizado é “`article`” não utilizamos as subdivisões que fariam sentido para livros e teses. As linhas com códigos novos foram comentadas:

```
\documentclass{article}
\usepackage{xltextra}
\title{Exemplo “LuTeX10a”: criando uma estrutura para seu texto}
\author{Lucas Mafaldo}
\date{\today}
\begin{document}
```

```
\maketitle
\section{Título da primeira seção} % Cria uma nova seção.
\subsection{Título da primeira subseção} % Cria uma seção subordinada à seção criada anteriormente.
    Conteúdo da \emph{primeira} subseção % O leitor ainda lembra do comando “\emph{}”?
\subsection{Título da \emph{segunda} subseção} % Podemos utilizar códigos de formatação nos títulos.
    Conteúdo da segunda subseção
\section{Nova seção}
    Conteúdo da nova seção.
\end{document}
```

No código acima, para efeitos de revisão, reunimos várias noções estudadas anteriormente. Temos agora um documento bastante completo, com informações textuais, título e várias subseções. Vamos ver como a aparência final do documento ficou:

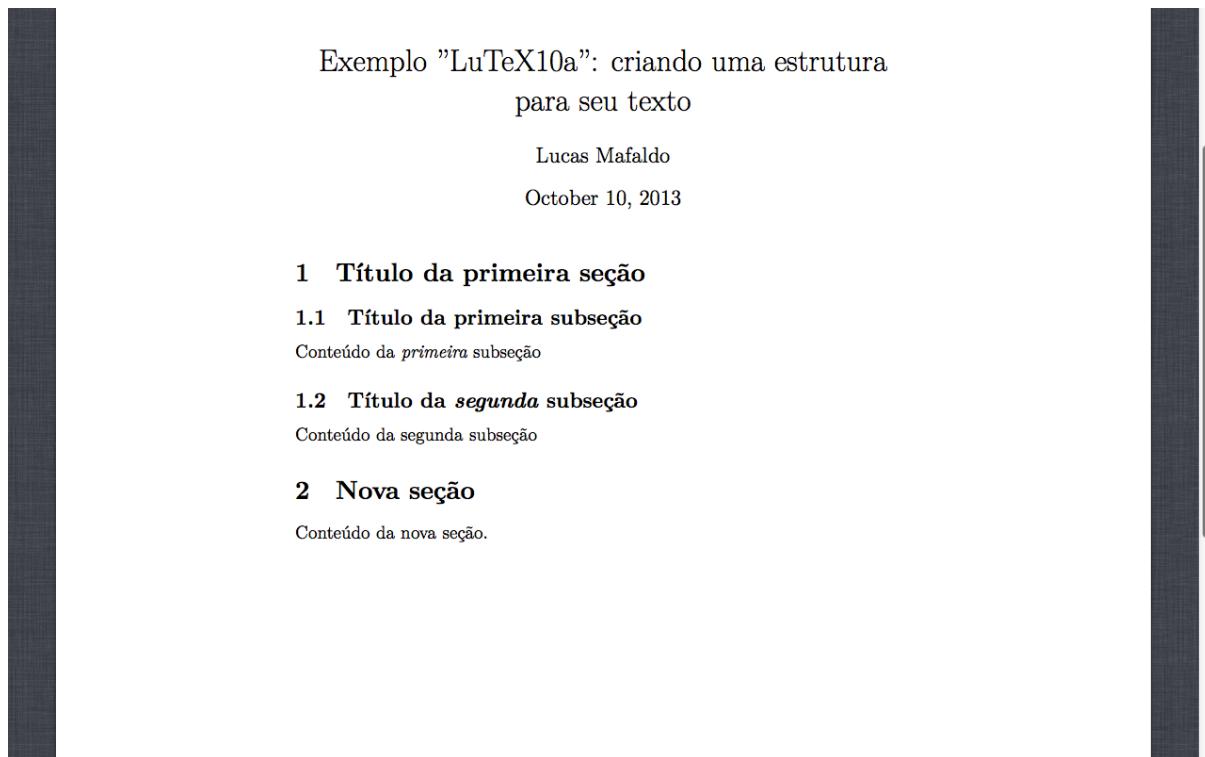


Imagen do arquivo “lutex10a.pdf”

Em termos comparativos, vamos modificar o documento para inserir alguns códigos de estrutura que só fazem sentido em um livro ou tese. Vamos alterar o código anterior, adicionar os comandos “\part” e “\chapter” e salvá-lo em um novo arquivo (lutex11a.tex). Novamente, nossas mudanças estão comentadas:

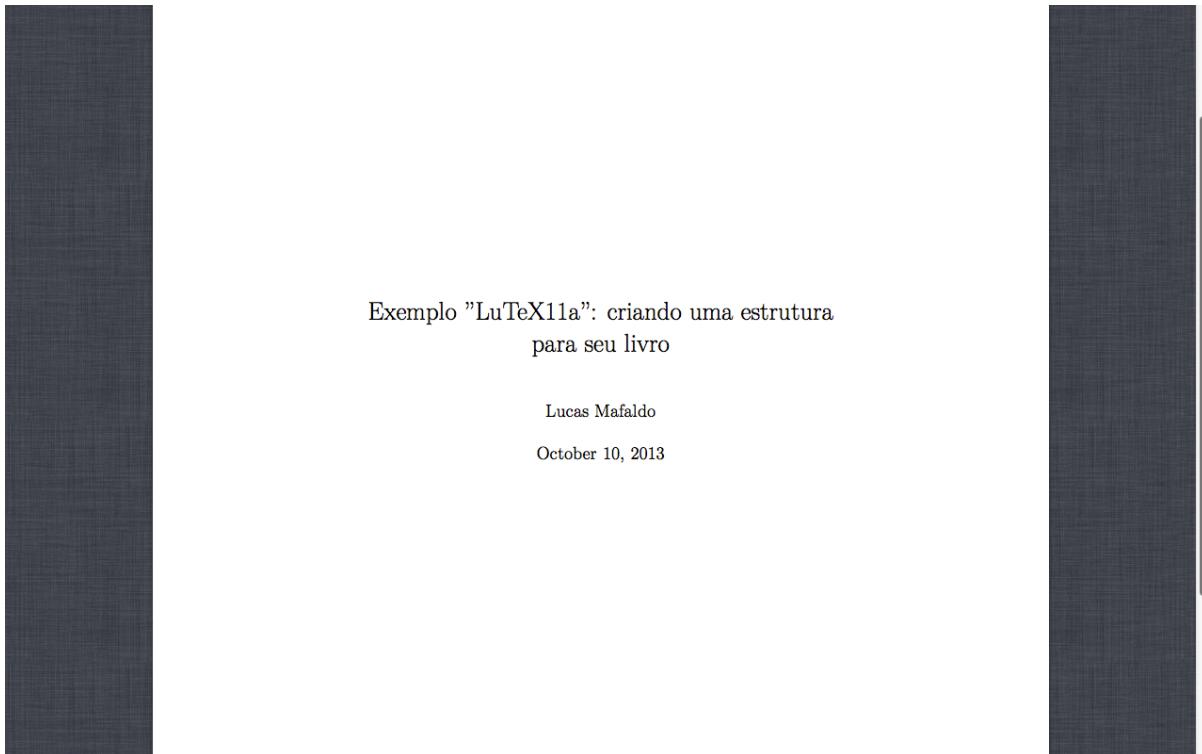
```
\documentclass[oneside]{book} % Modificamos essa linha, trocando “article” por “book” e adicionamos uma nova opção.  
\usepackage{xltxtra}  
\title{Exemplo “LuTeX11a”: criando uma estrutura para seu livro} % Novo título.  
\author{Lucas Mafaldo}  
\date{\today}  
\begin{document}  
\maketitle  
\part{Início do livro}  
\chapter{Conteúdo do arquivo LuTeX10a.tex}  
\section{Título da primeira seção} % Cria uma nova seção.  
\subsection{Título da primeira subseção} % Cria uma seção subordinada à seção criada anteriormente.  
Conteúdo da \emph{primeira} subseção % O leitor ainda lembra do comando “\emph{}”?  
\subsection{Título da \emph{segunda} subseção} % Podemos utilizar códigos de formatação nos títulos.  
Conteúdo da segunda subseção  
\section{Nova seção}  
Conteúdo da nova seção.  
\chapter{Capítulo com conteúdo novo}  
Conteúdo do segundo capítulo.  
\end{document}
```

Podemos ver que o nosso novo documento está consideravelmente mais longo. Primeiramente, vamos explicar porque adicionamos a opção “oneside” na classe do documento. Normalmente, o comportamento normal da classe “book” é preparar o livro para impressão frente e verso. Isso significa tanto criar margens diferentes dependendo do lado que a página será imprensa (para que haja mais espaço no ponto na parte interna da folha) como fazer com que cada capítulo comece em uma página direita – o que implica em deixar algumas folhas em branco. Como nosso plano não é imprimir o livro, mas apenas utilizarmos o arquivo no formato PDF, essa opção padrão não é interessante; e, por isso, a modificamos para a opção de imprimir apenas um lado da folha.

Os restantes dos novos comandos é bastante evidente. O comando “part” e “chapter” são utilizados do mesmo modo, com a opção de adicionar um título para cada uma dessas seções entre as chaves. Note-se que optamos por simplesmente repetir o conteúdo do primeiro capítulo colando o conteúdo do artigo que utilizamos no exemplo anterior. Uma das facilidades do LaTeX é justamente a complementariedade entre os comandos utilizados para criar a estrutura dos artigos

e dos livros. Note-se que é possível criar uma coletânea de artigos (ou um periódico acadêmico) praticamente sem esforço adicional: é fácil simplesmente adicionar o conteúdo dos artigos e gerar um documento mais longo.

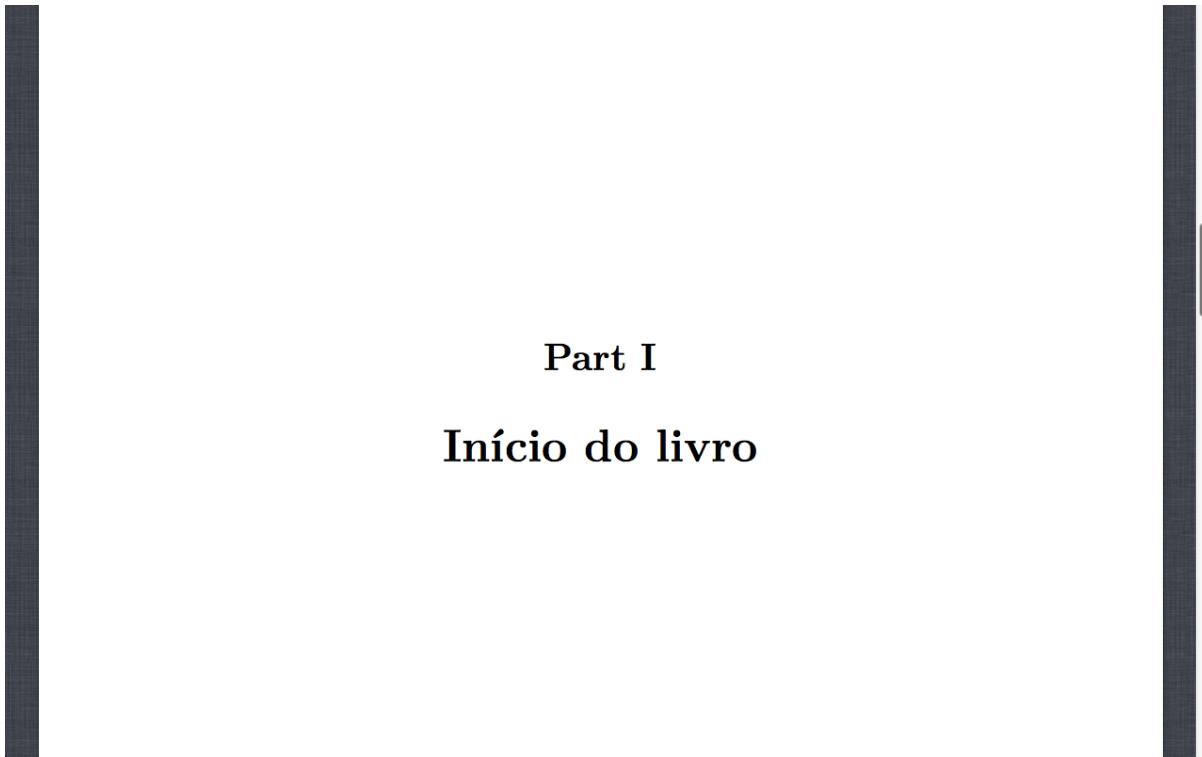
Vamos ver os resultados do nosso novo documento. Segue uma imagem da primeira página do arquivo “lutex11a.pdf”:



Primeira página do arquivo “lutex11a.pdf”

Como já deveríamos esperar, o LaTeX criou uma página exclusiva para o título por causa do comando “`\maketitle`”. No exemplo anterior isso não ocorreu porque a classe “`article`” coloca o texto na mesma página do título, enquanto a classe “`book`” cria uma página separada para cada coisa.

Vamos agora ver a segunda página do nosso documento:



## Part I

### Início do livro

Segunda página do arquivo “lutex11a.pdf”

Aprendemos uma nova informação sobre o LaTeX: a classe livro *também* coloca a seção “part” em uma página separada. No entanto, descobrimos também que não é muito interessante para nós: ele publica o nome “part” em inglês. O leitor certamente já adivinhou como resolver isso: podemos utilizar uma classe ou pacote que “ensine” ao LaTeX a colocar nosso documento em português. Nos próximos exemplos, faremos isso utilizando o pacote *polyglossia*. O leitor deve se sentir livre para tentar outra soluções.

Antes disso, no entanto, vamos verificar como está a terceira página do nosso documento:

# Chapter 1

## Conteúdo do arquivo LuTeX10a.tex

### 1.1 Título da primeira seção

#### 1.1.1 Título da primeira subseção

Conteúdo da *primeira* subseção

#### 1.1.2 Título da *segunda* subseção

Conteúdo da segunda subseção

### 1.2 Nova seção

Conteúdo da nova seção.

### Terceira página do arquivo “lutex11a.pdf”

Novamente, vemos que o LaTeX imprimiu tanto o título que escolhemos para esse capítulo, como avisou que esse era o “Chapter 1”. Obviamente, se estamos escrevendo um livro em português, esse é um comportamento que queremos modificar inserindo um dicionário que mude as convenções utilizadas pelo LaTeX. Aprendemos também algo novo sobre a classe “book”: enquanto a seção “part” ficava separada do seu conteúdo, em uma folha isolada, a seção “chapter” já começa diretamente com o conteúdo da sua seção.

O restante do documento irá seguir o comportamento esperado. É interessante, no entanto, reforçar algo implícito nas convenções adotadas pelo LaTeX: a seção “chapter” praticamente se comporta exatamente do mesmo modo que a classe “article” inteira. Nesse sentido, o LaTeX trata a classe “book” como se fosse uma coleção de “classes *articles*” identificadas pelo comando “chapter”. Essa característica é utilizada para facilitar a compilação de vários artigos em um livro maior.

No entanto, não é preciso se preocupar excessivamente com esses detalhes. O objetivo principal desta seção foi transmitir ao leitor os fundamentos do modo como o LaTeX utiliza a estrutura do seu texto. Novamente, vigora a separação entre *conteúdo* e *apresentação*: o leitor deve escrever seu texto pensando apenas na estrutura lógica do seu conteúdo e indicar ao LaTeX onde começa cada divisão. Posteriormente, na momento que for gerar o arquivo final, o LaTeX utilizará as convenções de cada classe para transmitir essa estrutura lógica dentro do padrão visual adequado para cada tipo de documento.

O único ponto a se lembrar que os seguintes comandos criam dividem seus textos em seções de modo hierárquico: cada uma das seguintes categorias maiores englobam todas as anteriores. Relembrando esses são os comandos que devem estruturar seu documento. A utilização deles é opcional, mas é interessante utilizar ao menos os comandos “\section” e “\subsection” em todos

os documentos e ao menos o comando “\chapter” em teses e livros.

1. \part{Nome da parte} % Comando opcional para livros e teses.
2. \chapter{Nome do capítulo} % Comando importante para livros e teses.
3. \section{Nome da seção} % Daqui em diante, esses comandos podem ser aplicados tanto para livros, teses como artigos.
4. \subsection{Nome da subseção}
5. \subsubsection{Nome da uma sub-subseção ainda mais interna}
6. \paragraph{Nome de subseção da sub-subseção}
7. \ subparagraph{Nome da sub-subseção da sub-subseção}

## O conteúdo do documento: gerando um sumário automaticamente

Entre as várias vantagens de organizar a estrutura do seu documento por meio de comandos para o LaTeX, está o fato de que gerar um sumário se tornará tão simples quanto inserir um único comando em seu documento. Basta colocar o comando “\tableofcontents” no *corpo* do seu documento que o LaTeX irá inserir no início do documento um sumário a partir dos comandos que vimos na seção anterior.

Como era de esperar, existem inúmeros modos de modificar a aparência e as características lógicas desse sumário. Você pode escolher o grau de profundidade (se deseja que apareça apenas os nomes dos capítulos ou também as seções e subseções). Além disso, você pode fazer com uma seção seja “escondida” do LaTeX no momento em que o sumário for gerado; para isso, basta adicionar um asterisco ao comando que indica o tipo de seção. Novamente, não iremos entrar em uma discussão exaustiva sobre *como* modificar seu sumário; iremos apenas ensinar os fundamentos do seu funcionamento e deixar para o leitor pesquisar funcionalidades adicionais.

Nesse sentido, precisamos explicar uma pequena complicação em relação ao modo como o LaTeX gera o sumário. Por incrível que pareça, é necessário gerar o mesmo arquivo *duas vezes* para que o sumário finalmente apareça no documento final. Isso acontece porque o LaTeX precisa primeiro fazer uma leitura completa do arquivo para gerar um arquivo temporário com a estrutura do seu documento. Depois, quando você rodar o mesmo arquivo *novamente* pelo LaTeX, ele irá utilizar esse arquivo temporário para imprimir o sumário no documento final. Embora isso pareça complicado, na prática é bastante simples: se você for adicionar um sumário ao documento, basta mandar seu editor gerar o arquivo “.pdf”, esperar o processo terminar e depois mandá-lo fazer a mesma coisa *novamente* (lembrando de *não* apagar os arquivos temporários entre os dois processos). Depois da segunda tentativa, seu documento estará do modo desejado.

Aliás, vale a pena avisar que esse não é o único caso onde é importante ativar o LaTeX repetidas vezes: o mesmo ocorre quando se utiliza um sistema de administração de referências bibliográficas. Na primeira vez, o LaTeX identifica as referências que serão utilizadas; na segunda, o sistema bibliográfico pega as informações específicas; e, apenas na terceira vez, o LaTeX pega as informações do sistema bibliográfico e as coloca no documento final. Note-se também que alguns editores, como o Sublime Text, fazem todos esses processos automaticamente, caso você tenha um plugin específico instalado. Em todo caso, uma dica prática que irá funcionar em todos

os editores é a seguinte: se alguma coisa deu errado em seu documento, tente simplesmente rodar o compilador novamente e veja se dessa vez dá certo.

Para tornar isso tudo menos abstrato, vamos voltar aos mesmos documentos do exemplo anterior e tentar adicionar o sumário a eles. Vamos aproveitar e adicionar também um pacote para avisar ao LaTeX que nosso documento será em português.

Na seção de exemplos, tomamos como base o arquivo “lutex10a.tex” utilizado como exemplo na seção anterior e o salvamos como “lutex10b.tex”. O leitor encontrará comentários adicionados às linhas que foram modificadas ou adicionadas. Em todo caso, eis o código da versão modificada do arquivo (lembmando que foi utilizado para gerar um documento na classe “article”):

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{polyglossia} % Novo pacote de línguas.
\setmainlanguage[brazil] % Especifica que a língua utilizada será o português brasileiro.
\title{Exemplo “LuTeX10b”: gerando um sumário a partir da estrutura do documento} % Novo título
\author{Lucas Mafaldo}
\date{\today}
\begin{document}
\maketitle
\tableofcontents % Comando que será utilizado para gerar o sumário.
\section{Título da primeira seção}
\subsection{Título da primeira subseção}
Conteúdo da \emph{primeira} subseção.
\subsection{Título da \emph{segunda} subseção}
Conteúdo da segunda subseção.
\section{Nova seção}
Conteúdo da nova seção.
\end{document}
```

Para mostrarmos como funciona a geração do sumário, vamos rodar esse arquivo uma única vez pelo TeXworks (lembmando de escolher a opção “XeLaTeX” como sistema de compilação ) e ver uma imagem do resultado:

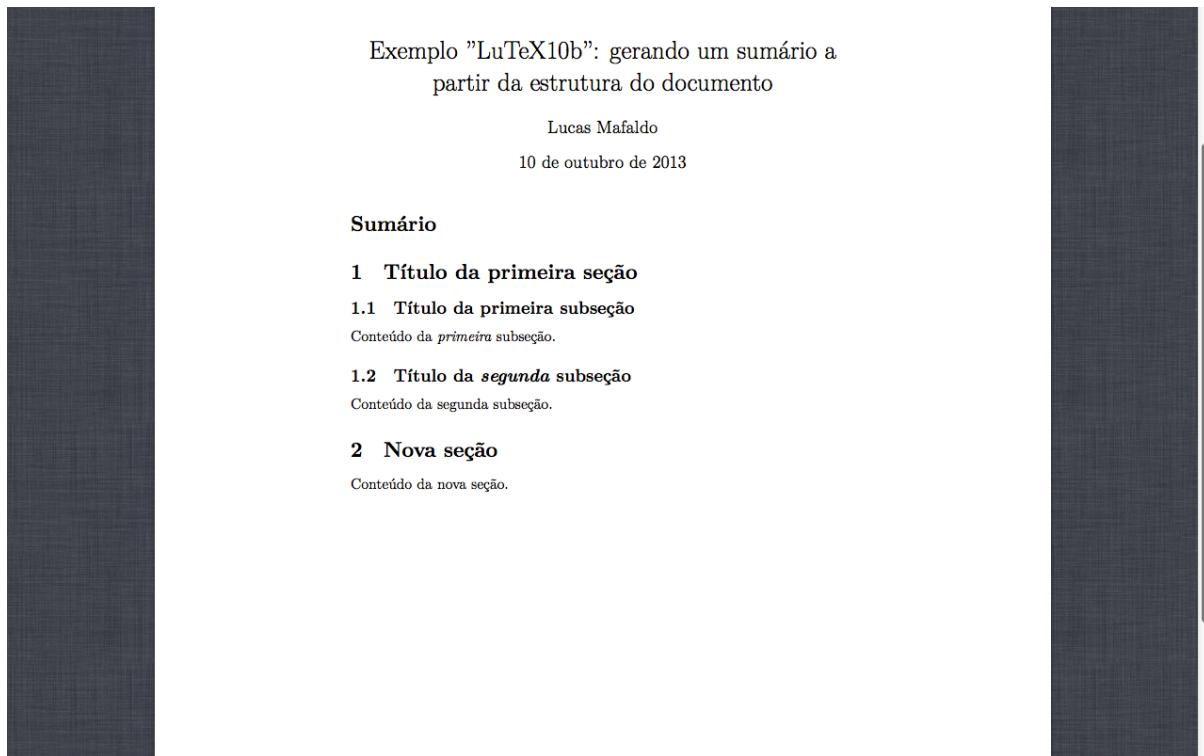


Imagen do arquivo “lutex10b.pdf” após uma única passagem pelo LaTeX

Primeiramente, vamos comparar esse exemplo com o resultado do mesmo arquivo na seção anterior onde *não* utilizamos o pacote “polyglossia”. Nesse novo exemplo, vimos o pacote modifcado para a nossa língua parte do conteúdo gerado automaticamente pelo LaTeX: ele colocou o termo “sumário” e colocou a data em português.

Tudo isso é ótimo, mas... a seção sumário está vazia. Depois dela, o documento começa imediatamente sem uma apresentação das seções do sumário. Complementarmente, se o leitor tiver feito o mesmo experimento, irá verificar que existe um arquivo temporário que foi gerado pelo LaTeX: agora há um arquivo chamado “lutex10b.toc” em seu diretório, cujo conteúdo irá dizer ao LaTeX o que colocar no lugar do sumário.

Novamente, estou dando ao leitor mais detalhes do que ele precisa aprender. Basta lembrar de *não* apagar nenhum arquivo temporário neste momento e mandar o TeXworks rodar o XeLaTeX *novamente*. Se fizer isso, o mesmo arquivo PDF será gerado novamente, mas dessa vez com o sumário correto, como podemos ver no seguinte exemplo:

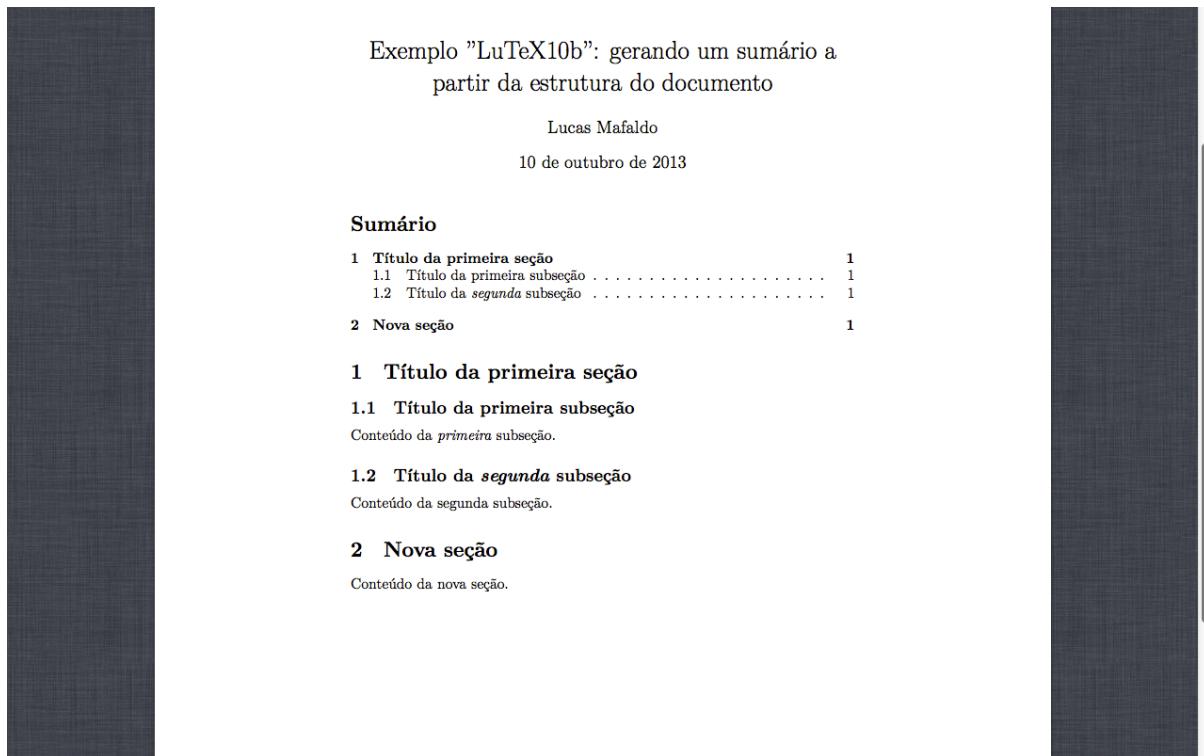


Imagen do arquivo “lutex10b.pdf” após passar *duas* vezes pelo LaTeX

Em suma, podemos ver que, apesar da explicação ser esquisita, o procedimento é bastante simples. Basta lembrar de gerar o arquivo *duas* vezes e de *não apagar* os arquivos temporários entre os dois processos. Note-se, aliás, que é interessante sempre lembrar de apagar os arquivos temporários *depois* que o processo final esteja concluído, já que eles não serão mais necessários e podem atrapalhar futuras compilações do mesmo documento. Em outras palavras: é uma boa política apagar os arquivos temporários *antes* ou *depois* de executar o processo de compilação, mas nunca *durante* a geração do arquivo PDF.

Para reforçar o conteúdo desta seção, vamos fazer as mesmas duas coisas com o nosso segundo exemplo da classe anterior: vamos pegar o arquivo “lutex11a.tex” e adicionar um pacote para passá-lo para o português e utilizar o comando “\tableofcontents” para gerar automaticamente um sumário. Dessa vez, não iremos publicar o novo código: o leitor já sabe o suficiente para realizar essas duas modificações por si mesmo. Depois de tentar fazê-lo por conta própria, compare seu resultado com o arquivo “lutex11b.tex” na seção de exemplos.

Dica: compare os arquivos “lutex10a.tex” e “lutex10b.tex” transcritos acima e observe suas diferenças. Lembre-se também de observar quais comandos devem ser adicionados ao preâmbulo quais devem ir para o corpo do documento.

Vamos, agora, comparar a diferença entre o arquivo “lutex11a.pdf” discutido na seção anterior e o arquivo “lutex11b.pdf” que será apresentado nas seções seguintes. Lembre-se das duas diferenças principais: adicionamos um pacote de língua e dissemos para o LaTeX gerar o sumário

automaticamente. Comecemos pela primeira página:

# Exemplo ”LuTeX11b”: gerando um sumário para seu livro

Lucas Mafaldo

10 de outubro de 2013

Imagen do arquivo “lutex11b.pdf”

Em relação ao exemplo anterior, a única diferença é que agora a data se encontra em português (graças ao *polyglossia*). Novamente, o fato de utilizarmos a classe “book” faz com o comando “\maketitle” gere um título em uma página separada. Vamos à segunda página:

# Sumário

Imagen da segunda página do arquivo “lutex11b.pdf” depois de ser gerado uma única vez

Ao contrário do que ocorreu com a classe “article”, o LaTeX dessa vez deixou o sumário em uma página separada (o que realmente tende a ocorrer com os livros). No entanto, temos aqui o mesmo problema: o conteúdo do sumário está vazio. Dessa vez já sabemos tanto o porquê como a solução: precisamos rodar o LaTeX novamente para que ele gere o sumário a partir dos seus arquivos temporários. Vamos fazer isso e ver o novo resultado:

# Sumário

<b>I Início do livro</b>	<b>2</b>
<b>1 Conteúdo do arquivo LuTeX10a.tex</b>	<b>3</b>
1.1 Título da primeira seção . . . . .	3
1.1.1 Título da primeira subseção . . . . .	3
1.1.2 Título da <i>segunda</i> subseção . . . . .	3
1.2 Nova seção . . . . .	3
<b>2 Capítulo com conteúdo novo</b>	<b>4</b>

Imagen da segunda página do arquivo “lutex11b.pdf” depois de ser gerado duas vezes

Dessa vez, nosso sumário apareceu corretamente. É interessante notar que várias convenções na geração do sumário mudaram em relação ao exemplo anterior feito com a classe “article”. Além do sumário estar em uma página separada, a numeração das seções também mudou, para incluir o capítulo na contagem das seções. Note-se que a seção relativa à “parte”, no caso, não participa dessa numeração: ela funciona como uma categoria mais geral, definida por numerais romanos que não afetam a numeração dos capítulos, das seções e subseções. Isso ocorre por ser inteiramente opcional dividir um livro ou tese em “partes”; essa categoria é tão ampla que só se faz necessário em obras especialmente longas e, por isso, o LaTeX prefere não considerá-la na numeração das outras seções. Note-se que a ausência da utilização do comando “\chapter” em uma classe de livro ou tese não seria nada interessante: a categoria capítulo é considerada pelo LaTeX como uma das características definidoras desse tipo de documento. Caso o leitor utilize a classe “book” e *não* faça referências ao capítulos, o LaTeX irá presumir que você está no “capítulo 0”, adicionando um zero antes da numeração das seções. Portanto, se não for a intenção do leitor utilizar as categorias “partes” e “capítulos”, o recomendável é utilizar outras classes (como “articles” ou inúmeras outras que podem ser encontradas em guias na internet).

Por fim, vamos ver como ficaram as terceiras e quartas folhas do nosso documento (já tínhamos vista essas versões no exemplo anterior, mas elas eram equivalentes à segunda e terceira folha do documento “lutex11a.tex”, já que essa versão tem uma folha adicional, devido ao sumário ocupar uma folha isolada).

# Parte I

## Início do livro

Imagen da terceira página do arquivo “lutex11b.pdf”

# Capítulo 1

## Conteúdo do arquivo LuTeX10a.tex

### 1.1 Título da primeira seção

#### 1.1.1 Título da primeira subseção

Conteúdo da *primeira* subseção.

#### 1.1.2 Título da *segunda* subseção

Conteúdo da segunda subseção.

### 1.2 Nova seção

Conteúdo da nova seção.

Imagen da quarta página do arquivo “lutex11b.pdf”

Comparando essa com a versão anterior, vemos que a única diferença significativa é que o pacote “polyglossia” traduziu para o português os termos “part” e “chapter”; em termos de aparência, no entanto, o documento permanece igual: a parte ainda está em uma página isolada, enquanto

o capítulo começa diretamente com o conteúdo. Novamente, isso serve para reforçar como um pacote de línguas funciona: ele mudou apenas o dicionário utilizado pelo LaTeX, deixando intacto as características visuais globais do documento. Se quiser, o leitor pode explorar como realizar essa mudança com outras classes e analisar as respectivas diferenças.

Em suma, vamos revisar os pontos principais discutidos nesta seção: 1. Os comandos que criam seções em seu documento (`\section`, `\chapter`, etc.) podem ser utilizados pelo LaTeX para gerar automaticamente um sumário do seu arquivo. 2. Para que isso aconteça, no entanto, é preciso “rodar” o arquivo do LaTeX *duas* vezes, pois apenas na segunda ele poderá acessar os arquivos temporários criados da primeira vez para imprimir corretamente o documento. 3. O título das seções terão uma aparência específica de acordo com a classe do documento. Classes distintas (como “book” e “article”) geram títulos e sumários distintos. 4. Podemos utilizar novas classes ou pacotes de língua para modificar a configuração e a aparência do sumário e dos títulos das seções (como fizemos com o *polyglossia* para mudar os termos “chapter” e “part” por “capítulo” e “parte”).

Por fim, é compreensível que o leitor eventualmente queira modificar a aparência ou qualquer outra característica do sumário gerado pelo LaTeX. Existem inúmeros comandos e pacotes que podem ajudá-lo a fazer isso. Como sempre, recomendamos que o leitor utilize o conteúdo desta seção como base para fazer pesquisas mais avançadas em busca de soluções específicas para suas necessidades pessoais. Em todo caso, o fundamental é que o autor do documento se lembre de – enquanto estiver redigindo seu texto – indicar a estrutura lógica por trás de cada passagem. Posteriormente, ele poderá modificar o preâmbulo para cuidar da aparência do sumário e da apresentação de cada parte do documento do modo como lhe parecer mais interessante.

## O conteúdo do documento: os comandos internos ao texto

Finalmente, nesta seção, iremos discutir os códigos que, de fato, irão compor o conteúdo dos nossos documentos. Note-se que, no LaTeX, é bem mais raro precisar ajustar a formatação do texto do que nos editores convencionais. Nos editores convencionais, precisamos fazer esses ajustes para colocar o texto em um padrão específico; no LaTeX, o texto por definição está no padrão desejado. Portanto, no LaTeX, só precisamos utilizar esses códigos quando queremos *fugir* dos padrões que nós mesmo estabelecemos para o documento. No preâmbulo, nós definimos todas as formatações desejadas; no corpo do texto, iremos apenas indicar quando queremos criar seções com formatações especiais.

Nesse sentido, em geral, quando utilizamos comandos do LaTeX dentro do corpo do documento, estamos querendo criar um ambiente de exceção ao padrão normal do texto. Nos primeiros capítulos, já vimos um exemplo de algo que precisa ser determinado apenas no próprio corpo do texto: o comando para enfatizar certas palavras (`\emph{}`). Como esse comando implica uma exceção em relação ao comportamento padrão, ele não pode ser previsto no preâmbulo. Do mesmo modo, sempre que precisarmos indicar que nosso texto terá um trecho que irá fugir ao comportamento normal do resto do documento, precisamos utilizar um comando específico do LaTeX.

Também aqui há diversos modos em realizar os mesmos objetivos. E também aqui os códigos dentro do texto interagem com os parâmetros estabelecidos no preâmbulo. Em geral, a maior

parte dos comandos utilizados *dentro* do texto funcionam criando um “ambiente” onde regras especiais devem ser aplicadas. Como sempre ocorre no LaTeX, existe um comportamento padrão para esses “ambientes” que podem ser modificados de acordo com a classe e com os pacotes utilizados.

Em certo sentido, todos esses ambientes funcionam de modo análogo aos comandos que dão início ao texto propriamente dito. Como já vimos, todo o conteúdo do nosso documento deve ficar contido nas linhas entre o comando “\begin{document}” e o comando “\end{document}”. Todo o texto entre esses comandos irá adquirir as características previstas a partir da classe e dos pacotes definidos no preâmbulo. Ou seja, em certo sentido, esse comando cria o grande ambiente “document” onde todo o conteúdo do seu texto deverá residir; além disso, esse grande ambiente irá seguir as regras gerais definidas pelo preâmbulo.

Todos os outros ambientes se comportam de modo análogos. Toda vez que o autor utiliza um comando “\begin{ambiente-especial}” ele está dizendo para o LaTeX que aquele trecho que começa ali e continua até aparecer o comando complementar (“\end{ambiente-especial}”) deve seguir regras especiais para ele. O LaTeX, novamente, irá procurar no preâmbulo as regras para lidar com esse ambiente de exceção.

Note-se que, mesmo ao criarmos esses ambientes especiais, o LaTeX ainda está operando sob o princípio da separação entre o conteúdo e a aparência do texto: o comando apenas indica a *lógica* por trás do ambiente especial (se é uma citação, nota de rodapé, etc.), mas o modo como esse ambiente será *apresentado* ao leitor dependerá inteiramente de características definidas no preâmbulo. Como sempre, a separação entre *forma* e *conteúdo* do texto permite ao autor modificar posteriormente as regras de cada um desses ambientes especiais, aumentando o controle do autor sobre seu documento.

## Citações

Um ambiente que interessará aos acadêmicos em particular é o ambiente utilizado para inserir citações em seu trabalho. O código padrão segue o seguinte modelo: “\begin{quote} ... (conteúdo da citação) ... \end{quote}”. Podemos ver a utilização desse ambiente no seguinte código, originário do arquivo de exemplo “lutex12.tex”:

```
\documentclass{article} % Alterne a classe entre “abntex2” e “article” e compare os resultados.  
\usepackage{xltxtra}  
\begin{document}  
Texto normal. Texto normal. Texto normal. Texto normal. Texto normal. Texto normal. Texto  
normal. Texto normal.  
\begin{quote}  
Texto citado. Texto citado. Texto citado. Texto citado. Texto citado. Texto citado.  
Texto citado. Texto citado. Texto citado. Texto citado.  
\end{quote}
```

```
\end{document}
```

Primeiramente, é importante notar que o ambiente especial “quote” deve sempre estar dentro do ambiente geral “document”. Note-se também que é possível colocar todo o “ambiente citação” em uma mesma linha, já que ele irá automaticamente criar um parágrafo separado para a citação.

```
\begin{quote} Texto citado. Texto citado. Texto citado. Texto citado. Texto citado. Texto citado.  
Texto citado. Texto citado. Texto citado. Texto citado. Texto citado. \end{quote}
```

Por fim, é sempre interessante observar na prática como funciona a separação que o LaTeX faz entre forma e conteúdo. No código acima, no corpo do documento, estamos basicamente dizendo para o LaTeX que o trecho citado é uma citação, mas *não* estamos dizemos *como* uma citação deve aparecer no documento. O LaTeX irá procurar essa orientação no preâmbulo e, na ausência de um comando ou pacote específico, ele irá utilizar o comportamento padrão da classe utilizada. Por isso, se mudarmos a classe, iremos mudar também o modo como a aparência da citação (e do resto do documento) será registrada no documento final.

Como exemplo, vamos comparar as duas imagens seguintes. Ambas foram resultados do mesmo código base, com a única diferença de termos alternado entre as classes “article” e “abnTeX2”.

Texto normal. Texto normal. Texto normal. Texto normal. Texto normal.  
Texto normal. Texto normal. Texto normal.

Texto citado. Texto citado. Texto citado. Texto citado. Texto  
citado. Texto citado. Texto citado. Texto citado. Texto citado.  
Texto citado. Texto citado. Texto citado.

Imagen do arquivo “lutex12a.pdf” (classe article)

Texto normal.  
Texto normal.

Texto citado. Texto citado. Texto citado. Texto citado. Texto citado. Texto citado. Texto  
citado. Texto citado. Texto citado. Texto citado. Texto citado. Texto citado.

Imagen do arquivo “lutex12b.pdf” (classe abnTeX2)

Embora tenhamos partido de um arquivo bastante simples, as diferenças entre as duas imagens são evidentes: tanto as margens da citação como de todo o texto são bem maiores na classe abnTeX2. Evidentemente, essa diferença não é acidental; independentemente de qual versão o leitor ache mais agradável em termos visuais, o ponto fundamental é que cada uma foi produzida a partir de normas textuais distintas. Em suma, o que o leitor deve lembrar em relação a esse comando é o seguinte: no momento em que estiver escrevendo, limite-se a indicar o ambiente apropriado para cada trecho; a configuração visual será feita posteriormente a partir do preâmbulo.

## Listas

Há inúmeros ambientes distintos que podem ser relevantes para o leitor. Existe, por exemplo, um ambiente para criar listas em seu documento. Se a lista for numerada, utilizaremos os códigos “`\begin{enumerate} ... \end{enumerate}`”; se for uma lista simples, utilizaremos o código “`\begin{itemize} ... \end{itemize}`”; é possível também utilizar uma lista para apresentar definições dos ítems com o código “`\begin{description} ... \end{description}`”. É possível também colocar listas dentro de listas, com alguma criatividade.

Em todo caso, *depois* de criar o ambiente da lista, o autor do documento deve utilizar um comando específico para indicar o início de cada ítem da lista. Cada ítem se comportará de acordo com as regras do ambiente (que, por sua vez, irá seguir as regras globais estabelecidas no preâmbulo). Em todos os casos, o comando “`\item`” inicia um novo ítem; se for utilizar um item para fazer uma descrição, é preciso também adicionar o nome a ser descrito com o seguinte comando: `\item[termo a ser descrito]`.

Para tornar isso mais claro, sugerimos que o leitor considere o seguinte código-fonte do documento “lutex13.tex” onde esses três tipos de lista são utilizado.

```
\documentclass{article}
\usepackage{xltextra}
\begin{document}
\begin{enumerate}
\item Primeiro ítem;
\item Segundo ítem.
\end{enumerate}
\begin{itemize}
\item Um ítem não-numerado.
\item Outro ítem não-numerado.
\end{itemize}
\begin{description}
\item[enumerate] é o código para criar listas numeradas.
\item[itemize] é o código para criar listas não-numeradas.
\item[description] é o código para criar listas descritivas.
\end{description}
\end{document}
```

Os códigos em questão são tão simples que o leitor certamente irá considerá-los auto-explicativos. Em todo caso, uma rápida olhada no documento final não deixará dúvidas sobre a função de cada um desses comandos:

1. Primeiro ítem;
2. Segundo ítem.
  - Um ítem não-numerado.
  - Outro ítem não-numerado

**enumerate** é o código para criar listas numeradas.

**itemize** é o código para criar listas não-numeradas.

**description** é o código para criar listas descritivas.

Imagen do arquivo “lutex13.pdf”

## Justificando trechos

É possível também utilizar ambientes (“`\begin{nome-do-ambiente}`” e “`\end{nome-do-ambiente}`”) para alinhar trechos do texto. Existem três possibilidades: alinhar o texto à esquerda (`flushleft`), à direita (`flushright`) ou no centro (`center`). O arquivo “`lutex14.tex`” traz um exemplo de como isso funciona na prática:

```
\documentclass{article}
\usepackage{xltxtra}
\begin{document}
\begin{flushleft}
Todo o texto desse ambiente ficará alinhado à esquerda.
\end{flushleft}
\begin{flushright}
Todo o texto desse ambiente ficará alinhado à direita.
\end{flushright}
\begin{center}
```

Todo o texto desse ambiente ficará centralizado.

```
\end{center}
```

```
\end{document}
```

Mais uma vez, esse código é tão intuitivo que não requer maiores explicações. Em todo caso, o leitor pode facilmente identificar como cada comando levou ao seguinte resultado final:

Todo o texto desse ambiente ficará alinhado à esquerda.

Todo o texto desse ambiente ficará alinhado à direita.

Todo o texto desse ambiente ficará centralizado.

Imagen do arquivo “lutex14.pdf”

## As notas de rodapé

O comando para adicionar notas de rodapé é ligeiramente diferente dos discutidos logo acima. Ele é mais parecido com o comando para ênfase, onde todo o texto das notas de rodapé devem ficar *dentro* do comando, isto é, entre as chaves. O comando para nota de rodapé é “\footnote{texto da nota de rodapé}”. Assim como já nos acostumamos a esperar, o LaTeX cuidará de todos os aspectos relativos à configuração visual das notas de rodapé e irá calcular sua localização ideal nas páginas dos nossos documentos.

O código relativo às notas de rodapé devem ser inseridos exatamente no local onde se deseja que a referência apareça. O LaTeX irá inserir nesse local uma marcação – cuja aparência, como sempre, será definida de acordo com as características da classe – e depois irá inserir o texto na mesma referência no local adequado (novamente, o que será calculado a partir das características classe). Novamente, o LaTeX é incrivelmente inteligente em calcular o espaço ideal das notas

de rodapé e irá se ocupar em numerá-las e manter sua aparência estável ao longo de todo o documento. Se o leitor já lidou com um documento longo em um editor convencional, certamente já passou muito tempo “brigando” com o editor para evitar que uma nota de rodapé empurrasse o início do capítulo muito para a frente ou para que ela não ficasse expandida demais antes da nova seção. Enfim, visualmente, as notas de rodapé podem dar muito trabalho. No entanto, como o LaTeX possui um sistema muito eficiente de calcular e distribuir o texto no espaço do documento, simplesmente é possível *esquecer* essa preocupação. Como sempre, cabe apenas ao leitor colocar as notas de rodapé no lugar onde ele deseja que as *referências* a ela apareçam e o LaTeX irá se ocupar em publicar seu conteúdo no lugar mais conveniente – como sempre, é claro, seguindo as determinações globais estabelecidas pelas classes e pacotes presentes no preâmbulo do documento.

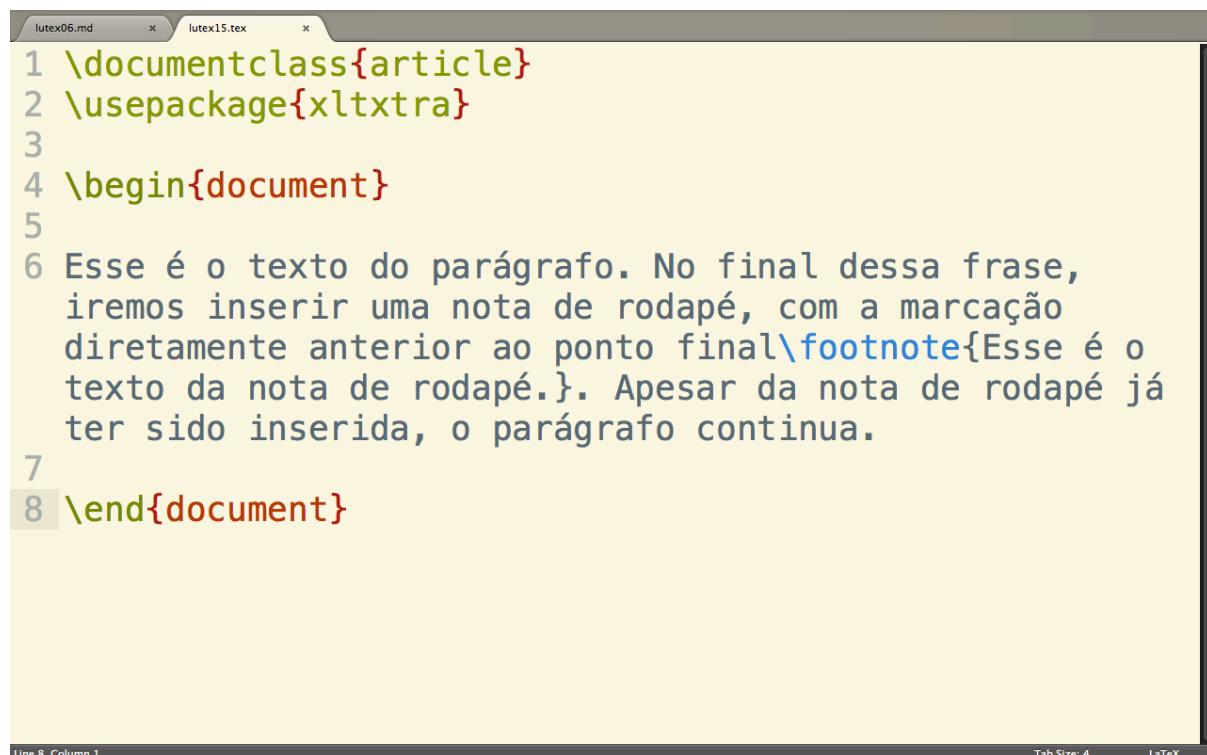
O único inconveniente do modo como o LaTeX utiliza as notas de rodapé é que algumas pessoas acham esquisito o modo como elas aparecem no código-fonte do texto; em outras palavras, elas reconhecem que o produto final fica perfeito, mas acham esquisito que, enquanto estão trabalhando no texto, tenham que ver as notas misturadas com o parágrafo em que elas são citadas. Para ficar mais claro o que estamos falando, vamos ver um exemplo de um texto com o código de uma nota de rodapé (referente ao arquivo “lutex15a.tex” na lista de exemplos).

```
\documentclass{article}
\usepackage{xltxtra}
\begin{document}

Esse é o texto do parágrafo. No final dessa frase, iremos inserir uma nota de rodapé, com
a marcação diretamente anterior ao ponto final\footnote{Esse é o texto da nota de rodapé.}.
Apesar da nota de rodapé já ter sido inserida, o parágrafo continua.

\end{document}
```

O leitor talvez tenha também sentido o incômodo de algumas pessoas: realmente é um pouco esquisito ver o código tão misturado com nosso texto. No entanto, alguns editores podem diminuir bastante esse inconveniente. Alguns colocam lado a lado o código-fonte em LaTeX (o que você está vendo acima) e o resultado final (o arquivo em formato PDF). Outros atribuem cores diferentes para o texto e para diferentes comandos para tornar mais fácil para o autor identificar cada elemento. Na minha experiência pessoal, essa opção é bastante aceitável. Note, por exemplo, como o mesmo código aparece no Sublime Text:



```

1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \begin{document}
5
6 Esse é o texto do parágrafo. No final dessa frase,
    iremos inserir uma nota de rodapé, com a marcação
    diretamente anterior ao ponto final\footnote{Esse é o
    texto da nota de rodapé.}. Apesar da nota de rodapé já
    ter sido inserida, o parágrafo continua.
7
8 \end{document}

```

Line 8, Column 1      Tab Size: 4      LaTeX

Imagen do código do arquivo “lutex15a.tex”

Não é uma solução perfeita, mas torna a leitura do código fonte ainda mais fácil. Em geral, os editores podem deixar essa diferença ainda mais destacada mudando também a cor da fonte do texto interno à nota de rodapé.

Note-se que também há outra solução para outra queixa comum em relação a esse comando: algumas pessoas reclamam que ele é muito longo e que toma tempo demais digitar esse comando para cada nota de rodapé. Existem duas soluções para esse problema. Dependendo do editor, você pode criar uma tecla de atalho para inserir o comando inteiro automaticamente (no entanto, como isso é uma solução específica ao editor e não ao LaTeX, não poderei discutir como fazê-lo; sugiro aos interessados consultar os guias de uso dos seus editores). Uma solução nativa ao LaTeX é utilizar o comando para redefinir o comando por algo mais curto.

O comando para fazer isso é bastante complicado, mas – felizmente – não precisa ser decorado. Encontrei esse modelo no excelente guia “LaTeX for Luddites”. A solução do autor para esse problema foi inserir o seguinte código no preâmbulo que – por motivos que não saberia explicar – cria um código mais curto (\fn) que possui o mesmo efeito que o código mais longo (\footnote) – que ainda tem o benefício colateral, no Sublime Text, de automaticamente mudar a cor interna da fonte.

O código é o seguinte:

```
\newcommand\fn[1]{\footnote{\#1}}
```

Vejamos agora como uma versão modificada do documento anterior aparece no Sublime Text:

```

1 \documentclass{article}
2 \usepackage{xltextra}
3
4 \newcommand\nota[1]{\footnote{#1}}
5
6 \begin{document}
7
8 Esse é o texto do parágrafo. No final dessa frase,
  iremos inserir uma nota de rodapé, com a marcação
  diretamente anterior ao ponto final\fn{Esse é o texto
  da nota de rodapé.}. Apesar da nota de rodapé já ter
  inserida, o parágrafo continua.
9
10 \end{document}

```

Line 10, Column 15      Tab Size: 4      LaTeX

Imagen do código do arquivo “lutex15b.tex”

Note-se que, nessa versão, além do código ter ficado mais curto (e fácil de lembrar), a cor do texto ficou ainda mais destacada. Note-se também que o leitor pode utilizar o mesmo código para escolher algo que ele pessoalmente ache mais fácil de lembrar. Por exemplo, caso ele queira algo mais intuitivo, como “\nota”, ele pode fazer a seguinte modificação no comando anterior:

```
\newcommand\nota[1]{\footnote{#1}}
```

O código acima irá substituir o comando “\footnote” pelo comando “\nota”. Note-se também que o comando “\newcommand”, embora – como o leitor certamente percebeu – seja um pouco complicado de utilizar, pode ser usado para definir inúmeros outros comandos que o leitor considere muito longo em digitar.

Em todo caso, o leitor certamente percebeu que o mecanismo interno da utilização das notas de rodapé é extremamente simples: basta utilizar o comando “\footnote{}” onde você quer que a referência apareça, colocar o conteúdo da nota entre as chaves e deixar que o LaTeX faça todos os cálculos de acordo com as características gerais do documento definidas no preâmbulo. Por fim, caso ele ache esse comando incômodo, há soluções para deixá-lo mais conveniente; seja criando atalhos no editor, seja com o comando de redefinição exemplificado acima.

Por fim, vejamos algumas imagens do resultado final do código que vimos acima. A primeira imagem se refere ao topo do documento, enquanto a segunda imagem se refere especificamente

à nota de rodapé (a fonte ficaria pequena demais se colocássemos uma imagem da página inteira).

Esse é o texto do parágrafo. No final dessa frase, iremos inserir uma nota de rodapé, com a marcação diretamente anterior ao ponto final<sup>1</sup>. Apesar da nota de rodapé já ter inserida, o parágrafo continua.

Imagen do arquivo “lutex15a.pdf”

---

<sup>1</sup>Esse é o texto da nota de rodapé.

Imagen do arquivo “lutex15a.pdf”

Apesar desses pequenos inconvenientes, devo dizer que considero que o modo como o LaTeX

lida com as notas de rodapé um dos maiores motivos para utilizar esse sistema. Depois de gastar horas e mais horas revisando inúmeras notas de rodapé na minha dissertação de mestrado (escrita em um editor convencional), fiquei espantado em descobrir que as centenas de notas da tese de doutorado estavam todas perfeitamente diagramadas e numeradas pelo LaTeX. Considero que essa estabilidade e praticidade mais do que compensam o pequeno inconveniente de inserir o texto das notas de rodapé no meio dos meus parágrafos enquanto estou escrevendo. O leitor certamente irá considerar que é um custo pequeno para pagar por notas de rodapé altamente estáveis e automaticamente diagramadas ao longo de todo o documento.

## Bônus: inserindo documentos dentro de outros documentos

Espero que a lista anterior contenha todos os comandos absolutamente essenciais para realizar a maior parte das tarefas de rotina no LaTeX. Dito isso, gostaria de recomendar mais um comando que, apesar de não ser absolutamente necessário, na minha experiência torna a rotina de trabalho com o LaTeX bem mais eficiente.

Trata-se do comando para inserir um documento *dentro* de outro documento. Embora isso pareça estranho, a ideia é bastante simples: já que o mesmo preâmbulo pode servir para inúmeros documentos, porque não criar um único preâmbulo e reutilizá-lo repetidas vezes?

O leitor deve ter percebido que boa parte do trabalho envolvido em criar um arquivo no LaTeX só precisa ser feito uma única vez. Por exemplo, depois que tivermos um preâmbulo adaptado para produzir um artigo acadêmico segundo as normas da ABNT, podemos repetir o mesmo preâmbulo indefinidamente – ou, ao menos, até que a ABNT proponha novas regras ou que você necessite de novas funcionalidades. Do mesmo modo, um único preâmbulo pode servir para inúmeras teses, artigos, relatórios e trabalhos acadêmicos.

Por isso, não faz o menor sentido reescrever o preâmbulo a cada vez que um novo projeto for iniciado. Evidentemente, é bastante fácil copiar e colar o preâmbulo em um novo artigo, mas há outra solução que, além de prevenir a repetição de esforço, traz alguns benefícios adicionais. Trata-se de utilizar um comando para inserir um arquivo externo *dentro* do corpo de um documento com um preâmbulo previamente preparado. Trata-se do comando “\include{nome do arquivo}”. Esse comando, como o nome indica, inclui um arquivo externo *dentro* de outro arquivo em LaTeX.

Existem diferentes modos de utilizar esse recurso e também existem diferenças sutis entre dois comandos semelhantes: “\include{nome do arquivo}” e “\input{nome do arquivo}”. Ao invés de entrar nessas sutilezas (o primeiro é mais indicado para arquivos maiores, que devem ser processados separadamente, além de incluir o conteúdo do arquivo em uma página separada), irei me limitar a descrever o modo como eu mesmo utilizo esses arquivos. Creio que servirá de base para o leitor criar seus próprios procedimentos na utilização do LaTeX.

Na minha opinião, faz sentido deixar o arquivo com o *conteúdo* do texto (a parte do código que fica *dentro* dos comandos que criam o ambiente “document”) em um diretório separado do arquivo do preâmbulo. Desse modo, tenho certeza que posso trabalhar à vontade no texto sem afetar o último preâmbulo estável. Por isso, utilizo um arquivo parecido com o código do seguinte modelo. Suponhamos que este é o arquivo “modelo-artigo.tex”.

```
\documentclass{article}
\usepackage{xltxtra}
\begin{document}
\include{projeto-x.tex}
\end{document}
```

O leitor certamente já conhece as convenções do LaTeX para presumir o que está acontecendo no código acima: temos um preâmbulo normal, assim como os comandos que iniciam e terminam o ambiente “document”. No entanto, a diferença é que o *conteúdo* do documento consiste apenas em um código que “inclui” nessa passagem o conteúdo de outro arquivo escrito na linguagem do LaTeX (“.tex”).

Esse outro artigo deve consistir apenas no *conteúdo* do texto, já que todo o preâmbulo e mesmo as indicações do início do documento já estão presentes nesse documento. Nesse sentido, o conteúdo do arquivo “projeto-x.tex” poderia ser apenas o seguinte:

```
\section{Primeira seção}
Texto da primeira seção.
\section{Segunda seção}
Texto da segunda seção.
```

Na prática, o que o LaTeX vai fazer vai ser combinar os dois arquivos e processar algo que seria equivalente ao seguinte código, onde o conteúdo do arquivo-alvo aparece inteiramente no lugar do comando “include” no arquivo-base:

```
\documentclass{article}
\usepackage{xltxtra}
\begin{document}
\section{Primeira seção}
Texto da primeira seção.
\section{Segunda seção}
Texto da segunda seção.
\end{document}
```

O leitor deve estar se perguntando qual a vantagem em ter os dois arquivos separados dos exemplos anteriores ao invés de ter um código unificado como no exemplo acima. Bom, essa vantagem depende inteiramente da complexidade do seu documento. Não faz muito sentido utilizar esse comando para uma arquivo tão curto como um anterior. No entanto, em se tratando de projetos maiores, com preâmbulos elaborados e múltiplos capítulos, esse comando torna bem mais fácil dividir o trabalho, permitindo se concentrar em cada parte.

Pessoalmente, gosto de deixar cada capítulo de livro ou tese em um arquivo “.tex” separado e sem preâmbulo. Isso me permite tratar cada capítulo como uma unidade e me concentrar apenas em seu conteúdo. Também não preciso nem pensar na ordem do capítulo, já que depois será preciso apenas mudar a ordem do comando “\include” no documento-fonte. Outra vantagem dessa divisão é que, como o documento fica mais curto, fica mais fácil navegar ao longo do texto.

Além disso, essa separação fornece mais segurança e um melhor controle das versões. Posso modificar meu preâmbulo com a segurança de que não irei apagar nada importante em um capítulo específico. Igualmente, posso trabalhar à vontade nos capítulos, sabendo que não estou alterando nada na configuração global do documento nem nos demais capítulos.

Note-se ainda que esse comando permite até mesmo manter o arquivo que será processado em um diretório diferente de onde estão os arquivos de texto. Também é algo que acho interessante fazer, pois torna mais difícil apagar um arquivo de texto por engano ao apagar os arquivos temporários. Além disso, você pode manter um único arquivo-fonte, com o preâmbulo estável, e ir simplesmente modificando o código “\include” para gerar novos livros e artigos.

Para utilizar o comando “\include” com documentos em outras pastas, basta adicionar o “endereço” do diretório onde o LaTeX deverá encontrar o arquivo. Ele faz isso sempre tomando como base o diretório onde o arquivo-fonte está salvo, presumindo que o arquivo-alvo estará no mesmo diretório. Se ele não estiver, você deve dizer para o LaTeX procurar um caminho relativo ao diretório do arquivo-fonte. Para ficar mais claro, observe esses seguintes exemplos:

```
\include{pasta-do-projeto/projeto-x.tex} % Nesse caso, o LaTeX procura pelo diretório “pasta-do-projeto” que esteja no mesmo diretório do arquivo inicial.  
\include{../projeto-x.tex} % Nesse caso, o LaTeX “sai” da pasta de origem e procura o arquivo “projeto-x.tex” no diretório mais geral.  
\include{../../projeto/projeto-x.tex} % Já nesse caso, o LaTeX não apenas sai da pasta de origem, como ele tenta entrar em outra pasta chamada “projeto” que estaria dentro dessa pasta mais geral; depois disso, ele entra nela e procura pelo arquivo-alvo nesse local.
```

Enfim, há inúmeras variações em torno de *como* encontrar o arquivo-alvo, mas o efeito será sempre o mesmo: ele irá incluir o código do arquivo-alvo dentro do código do arquivo-fonte e processá-los em conjunto.

Para fins de ilustração, vejamos um exemplo de outro código onde iremos produzir um documento bem mais longo: um livro. Nesse caso, para tornar o trabalho mais simples, salvamos cada capítulo em um arquivo separado sem preâmbulo: apenas com o conteúdo do texto e suas

divisões internas (nome do capítulo, seções, subseções, etc.). Depois, criamos um arquivo que irá “chamar” cada um desses capítulos e gerar o arquivo principal. Lembre-se que apenas nesse arquivo-fonte haverá a necessidade de escrever o preâmbulo.

Note-se que também incluímos nesse código o comando “\tableofcontents”, que irá gerar normalmente o sumário a partir do conteúdo de cada arquivo adicionado. Note-se também que é possível adicionar texto diretamente no documento-fonte se necessário. Como o leitor irá administrar e dividir seus arquivos é uma decisão puramente pessoal.

Enfim, eis um exemplo de como ficaria um arquivo-fonte que poderia servir de modelo para um livro.

```
\documentclass{book}
\usepackage{xltextra}
\begin{document}
\tableofcontents
\include{pasta-do-projeto/projetoz-cap1.tex}
\include{pasta-do-projeto/projetoz-cap2.tex}
\include{pasta-do-projeto/projetoz-cap3.tex}
\include{pasta-do-projeto/projetoz-cap4.tex}
\end{document}
```

Certamente o leitor já adivinhou o que está acontecendo nesse arquivo: ele irá “puxar” cada um dos capítulos individualmente e gerar um único arquivo final com os cinco capítulos na ordem correta. Note-se que optei também por colocar todos os arquivos desse projeto em uma pasta específica, para ficar mais fácil administrá-los. O mesmo arquivo poderia, portanto, ser reutilizado para gerar inúmeros livros, modificando apenas o endereço dentro do comando “\include”.

## Notas finais

Embora os exemplos que vimos ao longo desse capítulo tenham sido todos de documentos bastante simples, esperamos que o leitor tenha compreendido o fundamento por trás de cada comando; partindo desses princípios, ele poderá construir modelos para produzir documentos bem mais complexos. Em certo sentido, a partir desse ponto, o leitor já tem todas as informações necessárias para produzir qualquer documento em LaTeX. Dependendo das suas necessidades, recomendamos que comece o quanto antes a fazer seus próprios experimentos e identificar suas necessidades específicas.

Os dois capítulos seguintes irão explicar duas funcionalidades adicionais ao LaTeX que considero extremamente úteis, mas certamente não são de interesse universal. O próximo capítulo irá

mostrar como utilizar um sistema de administração de entradas bibliográficas em um documento produzido no LaTeX, enquanto capítulo seguinte irá explicar como o LaTeX pode ser utilizado também para produzir apresentações. Considero que ambas as funcionalidades são fundamentais para quem segue carreira acadêmica e, como o leitor verá, exige apenas pouquíssimo esforço para quem já dominou o conhecimento do LaTeX discutido até esse ponto.

Em todo caso, o leitor que se sentir tentado a pular os dois capítulos seguintes, encontrará ainda um breve capítulo final, com algumas sugestões para seus passos futuros, assim como uma seção com a transcrição do código-fonte dos modelos e exemplos utilizados ao longo deste livro.

# Capítulo 6: Referências bibliográficas no LaTeX

## Introdução

Certamente o leitor já percebeu que sempre existem diversos modos alternativos de realizar os mesmos objetivos no LaTeX. Portanto, quando levamos em conta que a administração bibliográfica representa um aspecto crucial do trabalho acadêmico, não é de surpreender que existam inúmeras ferramentas relacionadas ao tema no LaTeX.

É possível produzir bibliografias manualmente criando um “ambiente” com o comando “`\begin{thebibliography} ... \end{thebibliography}`” (um tipo de comando com o qual o leitor já está bastante familiarizado). No entanto, a grande vantagem em utilizar o LaTeX é justamente automatizar procedimentos repetitivos – e há poucas coisas tão repetitivas na rotina acadêmica quanto refazer novas listas de referências bibliográficas para cada novo texto.

Existem diferentes ferramentas para esta finalidade. Entre elas, a mais amplamente utilizada é o BibTeX. O BibTeX é uma ferramenta auxiliar para LaTeX que utiliza com base dois tipos de arquivos distintos: um arquivo do tipo “.bib” que consiste em uma base de dados bibliográficos e um arquivo do tipo “.bst” que consiste em uma folha de estilo que determina como a bibliografia deve ser apresentada no documento final. O BibTeX é um programa que combina essas dois arquivos – os dados biográficos e a folha de estilo da bibliografia – e informa ao LaTeX como construir a bibliografia final desejada.

É bastante difícil produzir uma folha de estilo bibliográfica por conta própria, mas, novamente, esse é um ponto onde a natureza aberta do LaTeX nos ajuda a poupar esforço: a equipe do abnTeX2 já preparou uma folha de estilo bibliográfica segundo as regras da ABNT para o BibTeX. Por esse motivo, nossa recomendação é que o modo mais prático de garantir que sua bibliografia esteja nas normas da ABNT, é utilizar o BibTeX em conjunto com os pacotes e classes criados pela equipe do abnTeX2. Em todo caso, mesmo para quem quiser seguir outras normas, o BibTeX é tão amplamente utilizado que é muito fácil encontrar folhas de estilo para as mais diferentes normas bibliográficas utilizadas ao redor.

No entanto, caso o leitor, por algum outro motivo, queira explorar sistemas bibliográficos alternativos, recomenda-se que estude a possibilidade de utilizar o pacote *biblatex*, que foi construído com a promessa de possuir melhor suporte para unicode (o que é bastante importante para nós, por motivos já explorados neste livro).

Em todo caso, apesar de poupar considerável esforço no médio prazo, como tudo em relação ao LaTeX, é preciso algum esforço inicial em *aprender* a utilizar o BibTeX. Este capítulo tem como objetivo deixar essa primeira etapa mais fácil.

## O LaTeX e as referências bibliográficas

Espero que, nesta altura do livro, o leitor já tenha imaginado que a inserção das referências bibliográficas em nosso documento irá seguir o nosso já batido princípio da separação a *aparência do texto* e seu *conteúdo*. Vamos aplicar esse princípio à questão da bibliografia. Quando você encontra as seguintes referências bibliográficas em um texto, o que você está vendo?

Gonzalez (1998).

GONZALEZ, 1998.

Nas duas linhas acima, temos dois modos diferentes de *apresentação*, mas em ambos os casos temos exatamente a mesma *informação*. Em um editor convencional, a cada vez que formos citar o mesmo livro do Gonzalez, teremos que verificar se temos *tanto* a informação correta, *como* se estamos utilizando a convenção correta das normas pertinentes para o trabalho em questão. Em cada citação, portanto, há a possibilidade de dois tipos erros (factuais e formais), o que significa que devemos revisar cada citação individualmente. Além disso, caso tenhamos que passar um artigo pronto para uma norma inteiramente diferente, teremos que ir modificando individualmente cada uma das referências bibliográficas do nosso texto.

É nesse ponto que a separação entre conteúdo e forma no LaTeX nos ajuda a lidar com a bibliografia. Enquanto estamos escrevendo, não devemos *optar* por uma convenção textual, devemos apenas assinalar para o LaTeX que, naquele ponto do texto, deve haver uma referência para uma entrada bibliográfica específica. Ao longo deste capítulo, veremos como fazer isso na prática. No entanto, o leitor já deve ter adivinhado o fundamento geral desse procedimento: vamos utilizar um comando que dirá ao LaTeX a *lógica* por trás daquela referência e ele utilizará um conjunto de convenções previamente estabelecidos para descobrir o melhor modo de expressá-la no documento final.

Para realizar essa separação entre dados e estilo, o LaTeX normalmente trabalha com arquivos adicionais ao seu documento em LaTeX. Um desses arquivos será o arquivo de dados bibliográficos (geralmente com uma terminação “.bib”), enquanto o outro arquivo será a folha de estilo bibliográfica (normalmente algo como “.bst”). Pode haver inúmeras variações em relação à folha de estilo (que pode ser incluída por classes, pacotes, previamente instalada no LaTeX ou baixada da internet posteriormente). Em relação à base dados (o arquivo com as *informações* bibliográficas), é possível também encontrar bases gigantescas na internet, mas é preferível que o autor do documento crie seus próprios arquivos, apenas com as referências bibliográficas que lhe interessam em particular.

Em todo caso, o fundamental é o seguinte: o leitor precisa trabalhar agora com três tipos de arquivos: 1. Um arquivo que trará única base de dados (.bib) que poderá ser reaproveitada por inúmeros arquivos; 2. Uma folha de estilo (.bst) que poderá opcionalmente ser instalado automaticamente pelo LaTeX 3. Seu documento propriamente dito (.tex).

Em suma, o LaTeX lida com as referências bibliográficas segundo os mesmos princípios que lida com os outros elementos textuais: separando a aparência do conteúdo. Já conhecemos as

vantagens dessa separação: mais flexibilidade e estabilidade em nosso documento. Mudar todo o estilo bibliográfico do texto se torna tão simples quanto modificar um único comando no preâmbulo. Além disso, isso nos libera da preocupação em verificar cada referência bibliográfica individualmente.

Entretanto, para fazer isso, o LaTeX precisa utilizar um sistema auxiliar externo (no caso dos exemplos desse livro, utilizaremos o BibTeX especificamente). Portanto, precisamos de vários procedimentos adicionais para colocar esse princípio em prática. Ao longo desse capítulo, tentaremos adquirir uma visão geral sob o uso desses sistemas.

## A delicada combinação entre LaTeX e BibTeX

Imagino que o leitor já esteja cansado em ser repetidamente apresentados a nomes extremamente parecidos. No entanto, esse é um exemplo onde essa distinção é importante. Tecnicamente falando, esse capítulo lhe ensinará a utilizar um programa *externo* ao LaTeX: o BibTeX. Embora essa utilização seja quase automática (e também possa ser feito dentro do seu editor de LaTeX), ela exige alguns passos bastante curiosos. Por isso, creio que é importante adquirir uma noção básica sobre a relação entre LaTeX e BibTeX, para compreender como eles podem trabalhar em conjunto.

O que acontece com a geração da bibliografia é bastante parecido com o que vimos no capítulo anterior em relação aos sumários. Como vimos, é preciso dizer ao LaTeX para gerar o mesmo arquivo *duas* vezes para que ele eventualmente gere um arquivo com o sumário correto. Com a bibliografia ocorrerá algo parecido, mas ainda mais complicado: precisamos ativar o LaTeX, ativar o BibTeX e depois ativar o LaTeX *duas* vezes novamente.

É possível simplesmente seguir esses passos automaticamente e tudo dará certo. No entanto, creio ser interessante compreender a lógica por trás desse comportamento esquisito. O que acontece é mais ou menos o seguinte: 1. Da primeira vez que o LaTeX tenta transformar nosso arquivo “.tex” em “.pdf”, ele varre todo o arquivo procurando algumas informações fundamentais e as transformam em arquivos temporários. Entre esses arquivos, estão as informações para montar o sumário e para construir a bibliografia. 2. Depois que esses arquivos temporários estão gerados, nós rodamos o BibTeX que irá utilizar esses arquivos como guia para encontrar as informações bibliográficas relevantes e depois irá incluí-las em novos arquivos temporários. 3. Por fim, rodamos o LaTeX *novamente* (alguns pacotes e guias recomendam rodar o LaTeX *duas* vezes seguidas nessa etapa final como garantia) para que todas as informações coletadas nos arquivos temporários estejam presentes no arquivo final.

O leitor certamente percebeu que, em todo esse processo, os arquivos temporários são importantíssimos. Por isso, é igualmente importante que o leitor *não* apague os arquivos *durante* o processo.

Dito isto, considero igualmente importante apagar os arquivos *antes* ou *depois* de começar a gerar um novo arquivo “.pdf”, para evitar conflitos entre os arquivos temporários anteriores e os novos. Digamos que você fez alguma modificação na bibliografia ou no sumário e tentou rodar o LaTeX novamente: em alguns casos, o LaTeX irá ter sucesso em escrever as novas informações *em cima* dos arquivos temporários anteriores; em outros casos, as diferenças entre os arquivos temporários podem causar problemas de compatibilidade de informações ou processos.

Enfim, embora tudo isso pareça bastante complicado, é possível resumir todo esse processo nas seguintes etapas bastante diretas. Lembre-se que seu editor de LaTeX provavelmente pode rodar diretamente tanto o BibTeX quanto o LaTeX nativamente. Alguns editores (como o Sublime Text) podem ser configurados para realizar todos esses processos automaticamente.

Em todo caso, independentemente do editor utilizado, basta que o leitor se lembre das seguintes etapas para evitar todos os problemas:

1. Apague os arquivos temporários da seção anterior e *não* toque mais nesses arquivos durante o processo.
2. Ative o XeLaTeX (ou o sistema tipográfico de sua preferência).
3. Ative o BibTeX.
4. Ative o XeLaTeX novamente.
5. Ative o XeLaTeX *novamente*.
6. Apague os arquivos temporários.
7. Tenha o cuidado em *não apagar* os arquivos de trabalho, que agora incluem os arquivos da base de dados (“.bib”), da folha de estilo bibliográfico (“.bst”) e, evidentemente, o código-fonte do texto (“.tex”) e o documento final (“.pdf”)

Um detalhe importante: é normal que algumas mensagens de erros apareçam nos processos intermediários. Elas se tornam relevantes se persistirem até a última etapa ou se forem críticas o suficiente para interromper o processo. Em todo caso, é importante interpretar as mensagens que indicam problemas que persistem até o documento; para isso, como sempre, pesquisas na internet tenderão a oferecer soluções aceitáveis.

Erros comuns nesse processo são: 1. Errar a ordem da compilação (trocar o BibTeX pelo LaTeX); 2. Esquecer de escolher o sistema tipográfico adequado (novamente, nesse livro, todos nossos exemplos são compilados pelo XeLaTeX); 3. Apagar novos arquivos temporários ou deixar arquivos temporários das seções anteriores.

Lembre-se de evitar esses erros e siga os passos acima e não deverá encontrar grandes problemas. Logo verá que, apesar de complicado na teoria, esses passos são bastante intuitivos na prática.

## O conceito de base de dados bibliográficos

O leitor certamente já teve a experiência de procurar inúmeras vezes as mesmas informações relativas aos mesmos livros e verificar uma por uma se a formatação de cada entrada bibliográfica está correta. Além disso, mesmo ao tentar poupar esforço e buscar essa informação em um trabalho anterior, o leitor ainda terá que verificar cada informação e cada formatação individualmente se quiser ter certeza de que está tudo correto (se não há uma editora no lugar da cidade, ou se o ano de publicação está no lugar certo, etc.).

É nesse momento que o princípio do LaTeX em separar a *informação* da *aparência* se revela particularmente interessante. Ao invés de organizar os dados bibliográficos segundo uma formatação específica (o que torna difícil a extração das informações relevantes), a base de dados organiza as mesmas informações segundo categorias específicas. Desse modo, temos a

garantia de que só precisaremos reunir essas informações uma única vez; afinal, ela estará permanentemente registrada em um formato de fácil acesso.

Em todo caso, como é comum em se tratando do LaTeX, é preciso fazer um investimento inicial que será amplamente recompensado posteriormente. Esse investimento começa criando um banco de dados bibliográficos. A decisão de como organizar esse banco cabe inteiramente a cada usuário. Há quem prefira criar um banco de dados gigantesco com todos seus interesses de pesquisa; há quem prefira criar vários pequenos bancos de dados para assuntos específicos.

Em todo caso, esses bancos de dados seguirão sempre a mesma estrutura interna. Eles basicamente consistem em um arquivo de texto simples com uma descrição de todas as informações pertinentes para cada entrada bibliográfica segundo uma série de categorias pré-determinadas pelo BibTeX. É possível criar esses arquivos manualmente, mas é um processo cansativo e bastante suscetível a erros. Portanto, é altamente recomendável que se utilize um programa de administração de referências. Pessoalmente, utilizo o BibDesk (disponível apenas para OS X), mas existem inúmeras alternativas para Linux e Windows (JabRef, Mendeley, etc.).

Independentemente do programa utilizado, o resultado será o mesmo: um arquivo do tipo “bib” que terá uma série de informações listadas. Para cada entrada bibliográfica, será preciso criar um código específico. O objetivo desse código será utilizar a entrada posteriormente como referência bibliográfica. Por isso, é interessante escolher um “apelido” para o livro ou artigo que seja fácil de lembrar posteriormente. Note-se que os editores trazem vários recursos para tornar essa memorização desnecessária (puxando toda a bibliografia em uma lista quando se utiliza o comando de citação, por exemplo). Normalmente, o “apelido” da entrada bibliográfica é escolhido a partir de alguma combinação entre o nome do autor, o título do livro e a data de publicação. No entanto, essa decisão fica puramente ao critério do criador da bibliografia.

Associada ao “apelido” de cada entrada bibliográfica, haverá todas as informações relevantes para a construção da seção de referências bibliográficas organizados segundo uma série de categorias pré-determinadas logicamente. Nesse ponto, é preciso seguir exatamente as convenções do BibTeX (e, por isso, é mais fácil utilizar um programa que faça isso automaticamente) para garantir que sua bibliografia seja processada corretamente.

Ao reunirmos todas as informações relativas a cada entrada neste arquivo, iremos criar uma base de dados que poderá ser “puxada” pelo LaTeX inúmeras vezes. Portanto, embora seja preciso algum esforço para criar a primeira versão, só será preciso registrar cada entrada bibliográfica uma única vez.

Por exemplo, eis o conteúdo do arquivo “bibliografia.bib” que utilizaremos como exemplo de base bibliográfica ao longo deste livro:

```
@book{gonzalez-dialectic,  
  Address = {Evanston, Illinois},  
  Author = {Francisco J. Gonzalez},  
  Publisher = {Northwestern University Press},  
  Title = {Dialectic and Dialogue: Plato's Practice of Philosophical Inquiry},
```

```
Year = {1998}}

@article{plato-rhetoric,
Author = {Edward Schiappa},
Journal = {The American Journal of Philology},
Number = {4},
Pages = {457-470},
Title = {Did Plato Coin Rhetorike},
Volume = {111},
Year = {1990}}
```

Se você utilizar um programa de administração de bibliografia (o que eu recomendo), não precisará decorar os códigos dessas categorias. Mas, mesmo nesse caso, é interessante entender o que está acontecendo. Vamos analisar as duas entradas acima.

O código “@book” indica que a primeira entrada trata de um livro, dizendo para o BibTeX quais informações esperar (se fosse um artigo, ele iria esperar informações diferentes, como número e volume do periódico onde foi publicado). Em seguida, está o “apelido” que criei para essa entrada (qualquer opção fácil de lembrar seria aceitável). Por fim, está a informação relevante, organizada segundo as categorias indicadas pelo BibTeX: título, autor, ano, editora, endereço.

Na segunda entrada, temos o código “@article”: esse código indica que o que se segue é um artigo acadêmico. Por isso, o BibTeX espera que informações sejam categorizadas de um modo diferente: temos também “autor” e “autor”, mas agora temos outros dados como “nome do periódico”, “páginas”, “número” e “volume.”

Em posse dessas informações, o BibTeX poderá reorganizar esses dados em inúmeros formatos diferentes; ou seja, depois de registrar esses dados uma única vez, você nunca precisará digitá-los novamente, pois o LaTeX irá automaticamente buscar esses dados sempre que alguns dos seus artigos os requisitarem (além de automaticamente adaptá-los ao estilo bibliográfico escolhido).

Seja manualmente, seja utilizando um programa específico, é interessante que cada pesquisador monte sua própria base de dados, com as informações relevantes para suas próprias pesquisas. Para isso, ele precisa criar um arquivo com a terminação “.bib” que siga exatamente a estrutura apresentada acima. Se o leitor quiser colocar os exemplos desse capítulo em prática, recomendamos que crie um arquivo bibliográfico para realizar os testes desse capítulo. Esse primeiro arquivo pode ser simplesmente o código acima salvo em um arquivo “.bib” ou o leitor pode aproveitar para começar a criar sua própria base de dados.

Nos exemplos ao longo deste capítulo, utilizaremos sempre o arquivo “bibliografia.bib” (O nome é opcional, mas a terminação “.bib” é necessária) que possui exatamente o código exposto acima. Esse arquivo será sempre deixado na mesma pasta onde o arquivo “.tex” que irá utilizá-lo está gravado (se o colocássemos em uma pasta diferente, seria preciso informar ao LaTeX em qual diretório encontrá-lo).

Em suma, o primeiro passo para utilizar o BibTeX para gerar suas bibliografias é construir

sua primeira base de dados. Evidentemente, não é necessário digitar todos os títulos que você porventura virá a utilizar. Um procedimento mais realista é começar com uma pequena base de dados e expandí-la aos poucos.

Note-se que o BibTeX segue as mesmas regras do LaTeX para lidar com caracteres especiais. Por isso, ocasionalmente sinais gráficos que representam comandos especiais geram problemas. Em particular, o LaTeX ocasionalmente impede a geração de entradas bibliográficas que tragam o caractere “&” – que é, infelizmente, bastante comum como parte do nome de editoras estrangeiras. Nesses casos, há uma solução bastante simples: adicionar a barra invertida antes desse termo (`\&`), como vimos há alguns capítulos atrás. Caso encontre erros análogos, vale a mesma regra que já vimos sobre erros no processamento de arquivos em LaTeX: interpretar a mensagem de erro e procurar discussões sobre problemas semelhantes na internet.

## Como utilizar o banco de dados bibliográficos

Nesse ponto, o leitor certamente já percebeu que uma grande vantagem desse sistema é a separação entre um artigo específico e a base de dados geral. Como sua base dados irá agora residir em um arquivo próprio, ela poderá ser reaproveitada em inúmeros documentos diferentes. Estritamente falando, a sua bibliografia não irá ficar *dentro* do seu documento, como ocorre nos editores convencionais. Ele ficará gravada em um arquivo próprio e será “chamada” pelo LaTeX sempre que você quiser utilizá-la em um documento específico. Portanto, assim como o LaTeX separa o conteúdo da apresentação, ele também separa os dados bibliográficos dos documentos que eventualmente o utilizam.

Entretanto, para que isso seja possível, você precisa se acostumar a lidar com vários arquivos distintos para gerar um único documento. É preciso ao menos três elementos para gerar um documento com referências bibliográficas completas: o arquivo onde está seu documento (geralmente algo como “exemplo.tex”), o arquivo de estilo bibliográfico (algo como “estilo-bibliografia bst”) e o arquivo onde sua base dados irá residir (algo como “bibliografia.bib”). Como afirmamos na introdução, o arquivo de estilo pode ser dispensado se alguma classe ou pacote instalá-lo automaticamente – o que geralmente é recomendado no caso de produções acadêmicas, onde você *não* deve criar seu estilo pessoal, mas seguir algum que seja partilhado por toda a comunidade acadêmica da qual faz parte.

Nesse sentido, se você utilizar um estilo padrão, você ainda precisará de dois arquivos distintos para produzir um único documento: o arquivo “.bib” onde está sua base de dados e o arquivo “.tex” onde está seu documento. Portanto, a questão é a seguinte: como fazer com que esses dois arquivos se comuniquem?

O primeiro passo é adicionar o seguinte comando ao seu arquivo “.tex” para que ele saiba qual bibliografia você quer utilizar e onde ele irá encontrá-la. Esse comando funciona de modo análogo ao que explicamos no capítulo anterior em relação ao comando “`\include`”: você pode colocar o local relativo ou o endereço completo do diretório onde seu arquivo está localizado. O jeito mais simples é deixar seu arquivo “.bib” na mesma pasta do seu arquivo “.tex” e simplesmente colocar o nome do arquivo com os dados bibliográficos.

Se fizer isso, basta adicionar ao *corpo* do seu documento o seguinte comando: “`\bibliography{nome-do- arquivo}`”. Esse comando convencionalmente é colocado ao final do texto, mas *antes* do

comando que encerra o ambiente que delimita o seu documento. O seguinte exemplo talvez torne isso mais claro:

```
\documentclass{article}
\usepackage{xltextra}
\begin{document}
Seu texto.

\bibliography{bibliografia} % Comando que diz ao LaTeX o nome do arquivo "bib" onde sua
                           % bibliografia está gravada.

\end{document}
```

Caso o leitor tente gerar o arquivo acima, ele verá que nenhuma seção bibliográfica terá aparecido na versão final. Isso não é um erro: o comando apenas informou ao LaTeX *qual* a bibliografia que deve ser utilizada; ainda falta dizermos ao LaTeX o que fazer com essa bibliografia.

## Escolhendo um estilo bibliográfico

O próximo passo na geração da sua bibliografia é associar seu arquivo a uma folha de estilo bibliográfico. Assim como em relação às *classes* dos seus documentos, você pode criar seu próprio estilo; no entanto, como é bastante trabalhoso fazer isso, é preferível baixar da internet uma folha de estilo pronta.

Existem inúmeras opções, mas é preciso prestar atenção em uma detalhe que geralmente confunde bastante os iniciante no LaTeX: cada folha de estilo bibliográfico requer um *pacote* próprio para funcionar. Portanto, ao escolher uma folha de estilo, você precisa verificar qual pacote irá utilizar (o que veremos na próxima seção). Não é necessário se preocupar com sua base de dados (o arquivo “.bib” onde estão as informações da sua bibliografia), pois ele – desde que tenha sido produzido de acordo com a estrutura recomendada para o BibTeX – funcionará com pacotes e folhas de estilo diferentes. No entanto, ocasionalmente você se depara com folhas de estilo que exigem um pacote específico.

Neste capítulo, iremos ensiná-los a utilizar dois pacotes bibliográficos diferentes e, por isso, vamos também recomendar duas folhas de estilo distintas. O primeiro é uma folha de estilo específica para o pacote *natbib*, enquanto o segundo é a folha de estilo que vem com a instalação do abnTeX2.

Para adicionar um arquivo de estilo bibliográfico, o comando é igualmente simples. Basta adicionar a seguinte linha no corpo do documento: `\bibliographystyle{nome-do-estilo}`. (geralmente, a convenção é adicioná-lo logo em seguida ao comando “`\bibliography`”).

Como já afirmamos anteriormente, os estilos bibliográficos funcionam como as classes dos documentos: o LaTeX já vem com algumas instaladas, mas é possível instalar novos estilos. Ao longo desse capítulo, iremos utilizar duas alternativas: o estilo “plainnat” e o estilo do abnTeX2.

O motivo dessa escolha está relacionado com os dois pacotes que iremos discutir nesse livro. Como vimos, cada folha de estilo funciona com pacotes e classes específicas. O leitor já está cansado de ouvir: existem inúmeros modos de realizar as mesmas funções no LaTeX. O mesmo ocorre até mesmo na utilização do sistema bibliográfico BibTeX. Essa diversidade de funções dependem dos pacotes utilizados em seu documento. Nesse livro, vamos nos encontrar em apenas duas possibilidade: (i) a utilização do pacote “natbib”, por ser um dos mais populares mundialmente; (ii) e a utilização do pacote bibliográfico do abnTeX2, por ser adaptado às novas vigentes em nosso país.

Como utilizaremos dois pacotes, utilizaremos também duas folhas de estilo: o “plainnat” é um estilo que funciona com o “natbib”. Para adicioná-lo, basta utilizar o comando “\bibliographystyle{plainnat}” ao corpo do nosso documento. Em relação ao estilo da abnTeX2, não é preciso adicioná-lo com nenhum comando. Em sua versão atual, ele é instalado automaticamente quando inserirmos o pacote bibliográfico do abnTeX2 em nosso preâmbulo.

Ao longo do restante do capítulo, veremos como utilizar ambas as opções, por considerarmos que irão cobrir a maior parte das necessidades do nosso leitor. Em todo caso, o leitor deve sempre se lembrar que existem uma infinidade de opções disponíveis que podem ser encontradas online.

## Pacotes bibliográficos: utilizando o natbib

O pacote “natbib” normalmente já vem instalado em sua distribuição do LaTeX. Por isso, para utilizá-lo, você precisa apenas adicionar o seguinte comando ao seu preâmbulo:

```
\usepackage{natbib}
```

O natbib, no entanto, exige uma folha de estilo com características específicas. Por exemplo, ele não irá funcionar se tentarmos utilizá-lo em conjunto com a folha de estilo do abnTeX2. Por isso, precisamos também adicionar *ao corpo* do documento o comando que indica um estilo compatível com o “natbib”. Como vimos na seção anterior, iremos utilizar a opção “plainnat” por já vir automaticamente instalada nas distribuições do LaTeX. Note-se que existem inúmeros outros estilos bibliográficos disponíveis para o natbib que podem ser facilmente encontrados na internet.

Nesse ponto, já temos os elementos para criar nosso primeiro documento com bibliografia. O arquivo seguinte é o código fonte do arquivo “lutex16a.tex” que está em nossos arquivos anexos. Para rodá-lo corretamente, o leitor deve salvá-lo na mesma pasta em que está o arquivo “bibliografia.bib”, que também está disponível na seção de anexos (e cujo conteúdo foi publicado em uma seção anterior). Segue, portanto, o código do documento “lutex16a.tex” com todos os comandos novos comentados.

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{natbib} % Comando que diz para o LaTeX carregar o pacote “natbib”.
\begin{document}
Seu texto.

\bibliography{bibliografia} % Dizemos ao LaTeX qual base de dados bibliográficos utilizar.
\bibliographystyle{plainnat} % Dizemos ao LaTeX para formatar a bibliografia segundo o estilo “plainnat”.
\end{document}
```

Embora esse modelo seja perfeitamente funcional, dependendo da folha de estilo pode aparecer uma mensagem curiosa: “Package natbib Warning: Empty ‘thebibliography’ environment”. Isso ocorre porque o comportamento padrão do natbib é colocar na lista de referência *apenas* as entradas bibliográficas que foram efetivamente citadas em seu texto.

Caso o leitor utilize exatamente a mesma folha de estilo que utilizamos (plainnat) nenhuma mensagem de erro irá aparecer, mas ainda assim o natbib não irá gerar nenhuma bibliografia pelo mesmo motivo: em se tratando de um artigo, o natbib considera que a seção de bibliografia deve conter apenas as *referências bibliográficas*; como nosso texto não fez referência a nenhuma entrada bibliográfica, ele não tem o que imprimir.

Portanto, vamos modificar esse documento e criar uma nova versão com comandos adicionais que digam para o natbib o que desejamos “imprimir” na seção bibliográfica. O natbib possui dois comandos básicos para introduzir referências em seu texto. O comando “\citet{nome da referência}” é utilizado para introduzir uma referência externo ao fluxo texto, como ocorre ao final das citações. Complementarmente, o comando “\citet{nome da referência}” é utilizado para introduzir a referência *dentro* do texto corrente. Ambos os comandos admitem ainda uma variável secundária para inserir páginas: \citet[p. “número da página”]{nome da referência}.

É importante lembrar que o nome da referência que deve ser utilizado nestes comandos é exatamente o código (ou “apelido”) que criamos para cada entrada bibliográfica em nossa base de dados. Caso o leitor volte algumas páginas atrás e verifique o código-fonte do arquivo “bibliografia.bib” verá que criamos os seguintes códigos para as duas entradas bibliográficas da nossa base de dados: “plato-rhetoric” e “gonzalez-dialectic”.

Novamente, alguns usuários questionam se isso é realmente uma economia de esforço, já que é necessário lembrar do código criado para cada entrada bibliográfica. Contra isso, novamente afirmamos que depende bastante da sua rotina de trabalho: além de ser possível criar “apelidos” mais facilmente memoráveis para cada entrada bibliográfica, os editores de LaTeX são flexíveis o suficiente para lhe lembrar de vários comandos e códigos necessários. No Sublime Text, por exemplo, com algumas configurações mais avançadas é preciso criar um atalho no teclado para acionar o comando bibliográfico que abre automaticamente uma lista com todas as entradas bibliográficas possíveis para cada arquivo bibliográfico.

Vamos agora modificar o arquivo anterior e introduzir esses comandos em nosso texto. Note-se que, no primeiro caso, nós *não* vamos citar páginas específicas, enquanto no segundo caso iremos citar especificamente as páginas às quais estamos nos referindo. Eis o código-fonte do arquivo “lutex16b.tex”. Ao invés de inserir comentários, iremos pedir que o leitor tente identificar as novidades:

Vejamos um exemplo prático da utilização deste comando.

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{natbib}
\begin{document}
É provável que Platão tenha inventado o termo “retórica”. \citet{plato-rhetoric}
\citet[p. 245-274]{gonzalez-dialectic} apresenta uma interpretação interessante sobre a
\emph{Carta Sétima} de Platão
\bibliography{bibliografia}
\bibliographystyle{plainnat}
\end{document}
```

Vejamos agora o resultado:

É provável que Platão tenha inventado o termo "retórica". [Schiappa, 1990] Gonzalez [1998, p. 245-274] apresenta uma interpretação interessante sobre a *Carta Sétima* de Platão

## References

Francisco J. Gonzalez. *Dialectic and Dialogue: Plato's Practice of Philosophical Inquiry*. Northwestern University Press, Evanston, Illinois, 1998.

Edward Schiappa. Did plato coin rhetorike. *The American Journal of Philology*, 111(4):457–470, 1990.

Imagen do arquivo gerado pelo código “lutex16b.tex”

Podemos observar que cada um dos comandos gerou um tipo de referência bibliográfica diferente; além disso, no segundo exemplo, adicionamos o número das páginas relevantes. Em ambos os casos, o LaTeX se encarregou de colocar as referências na formatação desejada e ainda gerou automaticamente uma lista de referências no estilo desejado.

No entanto, o leitor certamente já notou dois inconvenientes em nosso documento final. Primeiramente, o título da seção está em inglês (“References”). Além disso, o natbib está colocando na lista de referência apenas as obras que realmente foram citadas no documento e, embora esse seja o comportamento padrão, ocasionalmente nós queremos uma lista exaustiva da bibliografia consultada e não apenas daquelas que foram citadas. O leitor não precisa se preocupar: em se tratando de um pacote antigo e popular como o \_natbib, nossa velha máxima de que “há um comando para isso” certamente pode ser aplicada.

Espero que o leitor já tenha descoberto como resolver o primeiro problema: basta ensinar ao natbib como “falar” em português. Vamos introduzir o pacote “polyglossia” no preâmbulo e utilizá-lo para avisar ao LaTeX que nosso documento está escrito em português brasileiro. Não iremos repetir o código aqui, pois o leitor certamente já sabe como fazer isso por conta própria. Por isso, como exercício, pedimos que o leitor adicione ao arquivo anterior esse pacote e, depois, confira se ficou igual ao arquivo “latex16c.tex” que está na seção de anexos.

Se tiver feito tudo certo, o leitor deverá ter o seguinte resultado final, onde o termo “References” foi substituído por “Referências”

É provável que Platão tenha inventado o termo "retórica". [Schiappa, 1990] Gonzalez [1998, p. 245-274] apresenta uma interpretação interessante sobre a *Carta Sétima* de Platão

## Referências

Francisco J. Gonzalez. *Dialectic and Dialogue: Plato's Practice of Philosophical Inquiry*. Northwestern University Press, Evanston, Illinois, 1998.

Edward Schiappa. Did plato coin rhetorike. *The American Journal of Philology*, 111(4):457–470, 1990.

Imagen do arquivo gerado pelo código “lutex16c.tex”

Em relação ao seguinte problema, precisamos de um novo comando introduzido pelo pacote “natbib”: o comando “\nocite{}”. Esse comando pode ser utilizado para introduzir uma entrada bibliográfica específica ou para introduzir *todas* as entradas bibliográficas presentes em sua base de dados. No primeiro caso, basta colocar o código da referência entre as chaves; no segundo, para adicionar *todas* as referências presentes na base de dados, basta utilizar o comando “\nocite{\*}”.

Vamos mostrar os dois usos na prática. Primeiramente, vamos adicionar apenas uma nova entrada em um texto que não possui nenhuma referência. Segue o código do arquivo “lutex16d.tex”:

```
\documentclass{article}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage[brazil]
\usepackage[natbib]{natbib}
\begin{document}
Um texto sem referências.
\nocite{plato-rhetoric}
\bibliography{bibliografia}
\bibliographystyle{plainnat}
```

```
\end{document}
```

Espero que o leitor tenha notado que inserimos o pacote “polyglossia” nesse documento. Vamos agora ver o resultado final:

Um texto sem referências.

## Referências

Edward Schiappa. Did plato coin rhetorike. *The American Journal of Philology*, 111(4):457–470, 1990.

Imagen do arquivo gerado pelo código “lutex16d.tex”

Como podemos ver, embora o artigo não tenha sido citado, ele aparece na seção de referências. Portanto, este comando nos permite contornar o comportamento padrão do “natbib”, inserindo individualmente cada referência que queremos que apareça na bibliografia, embora não tenha sido citada ao longo do texto.

No entanto, é realmente bastante cansativo fazer esse procedimento em cada caso individual caso nossa bibliografia seja especialmente longa. Nesse caso, podemos utilizar o comando “\nocite{\*}” para imprimir todas entradas presentes no arquivo “bibliografia.bib”. Eis, como exemplo, o código do arquivo “lutex16e.tex”:

```
\documentclass{article}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage[brazil]
\usepackage{natbib}
```

```
\begin{document}  
Um texto sem referências.  
\nocite{*}  
\bibliography{bibliografia}  
\bibliographystyle{plainnat}  
\end{document}
```

Quando nos voltamos para o resultado final, veremos que todas as referências da nossa base de dados foram publicadas, mesmo sem terem sido citadas expressamente:

Um texto sem referências.

## Referências

Francisco J. Gonzalez. *Dialectic and Dialogue: Plato's Practice of Philosophical Inquiry*. Northwestern University Press, Evanston, Illinois, 1998.

Edward Schiappa. Did plato coin rhetorike. *The American Journal of Philology*, 111(4):457–470, 1990.

Imagen do arquivo gerado pelo código “lutex16e.tex”

Nesse ponto, o leitor já deve estar convencido que o LaTeX lhe permite modificar a aparência da sua bibliografia das maneiras mais diversas. Seguindo nosso tema ao longo deste livro, não iremos apresentar uma lista exaustiva dos comandos relativos à manipulação da bibliografia. O pacote natbib e o sistema BibTeX é tão amplamente utilizado que há uma quantidade vastíssima de recursos sobre o assunto na internet. Caso seja interesse do leitor, ele encontrará comandos, pacotes e estilos que mudarão a formatação, o estilo e os mais diversos aspectos da apresentação visual da seção bibliográfica. No entanto, caso queira fazer todas essas modificações simplesmente para enquadrar seus documentos nas regras da ABNT, existe um pacote que fará todas as mudanças necessárias ao mesmo tempo: ele será o tema da próxima seção.

## O pacote bibliográfico do abnTeX2

Caso o leitor precise seguir as regras da ABNT em seus documentos, repetimos nossa recomendação de que passe a utilizar os pacotes desenvolvidos pelo grupo abnTeX2. Se sua distribuição do LaTeX não for a mais recente, você precisará instalá-los separadamente. Recomendamos que visite a página do grupo para receber as informações mais atualizadas sobre como fazer isso:

<https://code.google.com/p/abntex2/><sup>6</sup>

A utilização do pacote do abnTeX2 é ainda mais simples que a utilização do natbib, mas com duas diferenças: (i) seu documento será produzido exatamente segundo as regras da ABNT; (ii) você irá utilizar automaticamente o estilo bibliográfico da ABNT e não poderá continuar utilizando os estilos do natbib (o que, convenhamos, é bastante razoável se seu objetivo for seguir as normas da ABNT).

Entretanto, não se trata de uma opção que precisa ser feita irrevogavelmente: sua mesma base de dados bibliográficos (seu arquivo “.bib”) irá funcionar com os dois pacotes. Portanto, é possível simplesmente alternar a utilização do pacote de acordo com as necessidades do seu documento. Se for produzir um documento segundo as regras da ABNT, certamente é mais fácil utilizar os pacotes do abnTeX2 do que tentar recriar a roda mexendo na configuração do natbib. No entanto, se você precisar adaptar um documento para outras normas, é só fazer algumas mudanças no preâmbulo, voltar a utilizar o “natbib” e procurar folhas de estilo apropriadas ao seu objetivo. Em suma, voltamos a uma das vantagens do LaTeX: a flexibilidade em poder reverter mudanças globais na aparência dos nossos documentos.

Em todo caso, vamos ver agora como utilizar o abnTeX2 para formatar nossas bibliografias. O pacote bibliográfico do abnTeX2 se chama “abnTeX2cite” e ele possui *dois* sistemas bibliográficos distintos: o sistema autor-data e o sistema numérico. Segundo a equipe do abnTeX2, ambos os sistemas são admitidos pelas normas da ABNT, mas ocasionalmente as instituições fazem opções específicas (o abnTeX2 recomenda o sistema numérico, mas alerta que muitas instituições preferem o sistema alfabético). Em todo caso, o pacote “abnTeX2cite” permite que o usuário escolha qual opção ele prefere. Para fizer isso, ele deve utilizar a opção “num” (para sistema numérico) ou a opção “alf” (para o sistema autor-data) como variável secundária no preâmbulo.

Portanto, se quiser utilizar o pacote *abnTeX2cite* no \_sistema numérico, você deve adicionar o seguinte código ao seu preâmbulo:

```
\usepackage[num]{abntex2cite}
```

Por outro lado, se quiser utilizar o pacote *abnTeX2cite* no \_sistema autor-data, você deve adicionar o seguinte código ao seu preâmbulo:

---

<sup>6</sup><https://code.google.com/p/abntex2/>

```
\usepackage[alf]{abntex2cite}
```

É importante notar uma diferença *muito importante* entre o *abnTeX2cite* e o *natbib*. No *natbib* você é obrigado a adicionar um estilo bibliográfico, enquanto no *abntex2cite* isso já é feito automaticamente. Portanto, se você alternar entre um pacote e outro é importante *remover* o código “\bibliographystyle” para evitar conflito entre folhas de estilos.

Outra diferença importante entre o *abnTeX2cite* e o *natbib* é em relação aos comandos utilizados para inserir as referências bibliográficas. No *abnTeX2cite*, para fazer uma citação *dentro* do texto se utiliza o comando “\citeonline[p.~”número da página]{nome da referência}”; enquanto para fazer uma referência *externa* ao texto se utiliza o comando “\cite[p.~”número da página]{nome da referência}”. Embora os comandos sejam parecidos, é necessário prestar atenção nas pequenas diferenças, senão o LaTeX não irá executar o comando corretamente. Note-se também que há uma diferença importante na indicação dos números de página: o *abnTeX2cite* pede explicitamente que você coloque o código “p.~” antes do número propriamente dito.

À título de comparação, vamos retornar ao arquivo que utilizamos com o *natbib* e adaptá-lo para o uso do *abnTeX2cite*. Em nossa folha de exemplo, esse é o arquivo “lutex17a.tex”:

```
\documentclass{article}
\usepackage{xltextra}
\usepackage[alf]{abntex2cite}
\begin{document}
É provável que Platão tenha inventado o termo “retórica”. \cite{plato-rhetoric}
\citeonline[p.~245-274]{gonzalez-dialectic} apresenta uma interpretação interessante sobre a
\emph{Carta Sétima} de Platão
\bibliography{bibliografia}
\end{document}
```

Compare o arquivo acima com o modelo lutex16b.tex. As mudanças começam no pacote utilizado, continuam com novos comandos e terminam com a importante *ausência* da folha de estilo (que já foi carregada automaticamente pelo *abnTeX2cite*).

Se nos voltarmos para o documento final, teremos o seguinte resultado.

É provável que Platão tenha inventado o termo "retórica". (SCHIAPPA, 1990)

Gonzalez (1998, p. 245-274) apresenta uma interpretação interessante sobre a *Carta Sétima* de Platão

## References

GONZALEZ, F. J. *Dialectic and Dialogue: Plato's Practice of Philosophical Inquiry*. Evanston, Illinois: Northwestern University Press, 1998.

SCHIAPPA, E. Did plato coin rhetorike. *The American Journal of Philology*, 1990. v. 111, n. 4, p. 457–470, 1990.

Imagen do arquivo gerado pelo código “lutex17a.tex”

Sugerimos o leitor volte um pouco no livro e compare as imagens dos arquivos “lutex16b.pdf” e “lutex17a.pdf” para observar o poder do LaTeX em organizar nossas referências bibliográficas. Em ambos os casos, o texto está exatamente o mesmo e utilizamos comandos bem parecidos (que podem facilmente serem redefinidos caso o leitor queira mudar de pacote); em ambos os casos, no entanto, os resultados estão sutilmente diferentes – o tipo de sutileza, no entanto, que é importantíssimo para as bancas que avaliam as normas dos trabalhos acadêmicos. Observem as seguintes mudanças:

Citação no texto (natbib):

Gonzalez [1998, p. 245-274]

Citação no texto (abnTeX2cite):

Gonzalez (1998, p. 245-274)

Citação na lista de referências (natbib):

Francisco J. Gonzalez.

Citação na lista de referências (abnTeX2cite):

GONZALEZ, F. J.

Esses pequenos detalhes, justamente por serem tão sutis, são extremamente difíceis de verificar individualmente. No entanto, eis que o LaTeX nos liberta da necessidade de cuidar disso: ele automaticamente adapta o mesmo texto às mais diferentes normas bibliográficas.

Note-se que reencontramos aqui o pequeno inconveniente do título da seção bibliográfica estar em inglês. Podemos resolver isso como fizemos em relação ao natbib (utilizando o polyglossia), mas já que queremos seguir as normas da ABNT e já temos o abnTeX2 instalados, temos acesso a uma solução bem mais conveniente: podemos modificar a *classe* do nosso documento para abnTeX2, para colocar tanto nosso documento em português, como colocar também o restante do texto (e não apenas as referências bibliográficas) nas regras da ABNT.

Vamos ver como fica o novo código (lutex17b.tex):

```
\documentclass[article]{abntex2} % Observem que “article” é uma opção da classe “abntex2”.
\usepackage{xltextra}
\usepackage[alf]{abntex2cite}
\begin{document}
É provável que Platão tenha inventado o termo “retórica”. \cite{plato-rhetoric}
\citetitle[p.~245-274]{gonzalez-dialectic} apresenta uma interpretação interessante sobre a
\emph{Carta Sétima} de Platão
\bibliography{bibliografia}
\end{document}
```

Existem um elemento que pode causar confusão neste código: até esse momento, estávamos utilizando “article” entre as chaves para designar uma classe e agora a mesma palavra reaparece aqui entre os colchetes. Apesar da semelhança, se trata de um uso totalmente diferente da mesma terma. Nos primeiros exemplos, “article” era utilizado entre chaves por se tratar de uma das *classes* padrões do LaTeX. Aqui ela é utilizada entre colchetes por ser uma das *opções* criadas pela classe abnTeX2. Portanto, na nossa primeira linha de comando, temos a utilização de uma classe inteiramente nova (a classe “abntex2”) que cria uma *nova opção* utilizada para produzir

artigos acadêmicos segundo as regras da ABNT. Portanto, não se deixe enganar pela semelhança entre esses códigos: apenas de utilizar a mesma palavra, trata-se de conceitos – e, portanto, de comandos – inteiramente diferentes.

Em todo caso, vamos observar o resultado:

É provável que Platão tenha inventado o termo "retórica". (SCHIAPPA, 1990)

Gonzalez (1998, p. 245-274) apresenta uma interpretação interessante sobre a *Carta Sétima* de Platão

## Referências

GONZALEZ, F. J. *Dialectic and Dialogue: Plato's Practice of Philosophical Inquiry*. Evanston, Illinois: Northwestern University Press, 1998.

SCHIAPPA, E. Did plato coin rhetorike. *The American Journal of Philology*, 1990. v. 111, n. 4, p. 457–470, 1990.

Imagen do arquivo gerado pelo código “lutex17b.tex”

De fato, quando nos voltamos para o resultado final, veremos que essa nova classe muda todo o documento: as margens estão reduzidas, o título da seção passou para o português e está agora centralizado. Essa última mudança ilustra a diferença entre utilizar uma classe (como o abntex2) e um pacote (como o polyglossia) para resolver o mesmo problema: o polyglossia tinha mudado apenas a língua do documento, enquanto o abntex2 muda todas suas características visuais para deixá-lo de acordo com as normas da ABNT.

Em todo caso, se o objetivo for colocar todo o documento nas normas da ABNT, não consigo imaginar um jeito mais prático do que fazê-lo do que utilizar em conjunto a classe e o pacote bibliográfico desenvolvido pela equipe abnTeX2. Recomendamos que o leitor procure a documentação própria para aprender os diferentes comandos necessários para produzir uma ampla variedade de documentos (relatórios, teses, artigos, livros, etc.).

## Notas finais

Embora o leitor tenha sido exposto a um volume considerável de informação, espero que tenha percebido que não é tão complicado descobrir como colocá-las em prática. Em relação à administração bibliográfica, os pontos centrais a se lembrar são os seguintes:

1. O leitor pode criar “ambientes” bibliográficos manualmente, mas é mais interessante utilizar um programa auxiliar para automatizar esse trabalho.
2. Nesse capítulo, recomendamos a utilização do BibTeX, por ser amplamente utilizado e possuir vários recursos adicionais.
3. Para utilizar o BibTeX, é preciso possuir um arquivo de dados bibliográficos (.bib) com todas as informações das entradas bibliográficas organizadas de modo lógico.
4. Recomendamos que utilizem um programa para administrar a bibliografia (BibDesk, JabRef, Mendeley, etc.).
5. É preciso também escolher um estilo bibliográfico; eles podem ser baixados da internet, automaticamente ativados por alguns pacotes (abnTeX2) ou pode-se utilizar os que já vêm instalados com o LaTeX.
6. É preciso também escolher um pacote bibliográfico para ensinar ao LaTeX como lidar com a bibliografia. Discutimos a utilização de dois pacotes: abnTeX2cite e natbib.
7. Para garantir que o LaTeX irá processar corretamente seu arquivo e incluir os dados bibliográficos é interessante seguir os seguintes passos exatamente nessa sequência (em alguns casos, seu editor pode fazer isso automaticamente ou, pelo menos, reduzir esses processos a atalho no teclado):
  1. Apagar os arquivos temporários anteriores;
  2. Rodar o XeLaTeX;
  3. Rodar o BibTeX;
  4. Rodar o XeLaTeX;
  5. Rodar o XeLaTeX novamente.

Depois desses passos, o leitor verá que seu investimento de tempo no LaTeX será enormemente compensado: ele agora possui um sistema bibliográfico que poderá ser reutilizado em todos os seus trabalhos futuros, simplificando enormemente a dificuldade em lidar com diversas normas bibliográficas.

Nesse ponto, o leitor já possui todas as ferramentas necessárias para utilizar o LaTeX para produzir qualquer tipo de texto. Dito isso, o LaTeX pode fazer *algo mais* do que produzir textos: no próximo capítulo, veremos como utilizar o LaTeX para preparar *apresentações*.

# Capítulo 7: Criando apresentações com o LaTeX

## Introdução

Rigorosamente falando, creio que o leitor já aprendeu tudo necessário para produzir um *texto acadêmico*. No entanto, existe outra função do LaTeX menos conhecida: criar apresentações. Talvez o leitor não saiba, mas apresentações não precisam ser feitas apenas em *Power Point* ou no *Open Office*. É possível também criar apresentações em formato “.pdf” com todos recursos básicos desse tipo de documento: animações, imagens, cores, listas, etc.

Embora pareça estranho utilizar o LaTeX para essa finalidade, reencontramos aqui as mesmas vantagens gerais do LaTeX: teremos mais controle sobre a aparência final do nosso documento e ele será consideravelmente mais estável. Além disso, não precisaremos mais nos preocupar em qual versão do Power Point nosso documento será visto, pois o documento final também pode ser exportado para o formato PDF. Por fim, uma vez estabelecido o modelo básico, podemos reutilizá-lo indefinidamente.

Dito isto, reconheço que nem todas as vantagens que o LaTeX possui no campo dos textos é igualmente transponível para o campo das apresentações. Em muitos casos, os autores das apresentações preferem inovar no campo visual, enquanto o LaTeX valoriza justamente a funcionalidade e estabilidade da aparência. Em outras palavras: a questão não é que as apresentações do LaTeX sejam mais ou menos agradáveis do ponto de vista estético; no LaTeX, haverá mais controle e precisão sobre a aparência final. Isso significa que é necessário um investimento maior no início que só será recompensado com a reutilização do modelo do documento.

Nesse sentido, para quem deseja ficar continuamente mudando sua apresentação em termos visuais, para quem quer ter a apresentação mais *criativa* possível, talvez o LaTeX realmente não seja o modelo ideal para essa finalidade. Como vimos ao longo do livro, o LaTeX é muito bom em criar um modelo e repetir esse modelo indefinidamente. Se alguém quiser que cada apresentação sua seja inteiramente diferente, será mais realmente cansativo continuamente criar novos modelos no LaTeX. (Note-se é facilíssimo alternar entre centenas de temas que já estão prontos na internet – literalmente, trata-se de mudar apenas uma única linha de código –; o que é difícil é criar temas inteiramente novos).

Entretanto, para o público cujo interesse é se concentrar no *conteúdo* da apresentação ou manter uma identidade visual estável ao longo de várias apresentações (ao longo de um evento ou em várias apresentações de uma mesma instituição), diríamos que o LaTeX é decididamente a opção superior *mesmo* no campo visual e estético. Para conferências e palestras onde o conteúdo é importante, notamos que é bastante comum que o palestrante utilize recursos que causam mais distração do que ajudam na condução da palestra. Como os editores de apresentações trazem o

foco para a *adição* de recursos visuais, as apresentações criadas por eles arriscam ter elementos *demais*, causando a perda o fio condutor da palestra.

O LaTeX, como vimos, nos forçar a pensar em nossos documentos em termos estruturais – e isso também ocorre nas apresentações. Alguns modelos de apresentações no LaTeX vem mesmo com guias visuais para identificarmos se estamos em uma seção maior ou em uma subseção de outro tópico, ajudando a plateia à acompanhar o raciocínio do expositor. Além disso, como o LaTeX oferece controle total sobre a aparência final do documento, é possível criar apresentações bastante interessantes em termos visuais – o limite, evidentemente, é a quantidade de tempo e esforço que o leitor quer investir nisso. Quem tiver interesse, pode gastar bastante tempo aperfeiçoando a identidade visual das suas apresentações e, depois de criar o modelo ideal, repetí-lo indefinidamente no futuro.

Certamente o público que irá aproveitar ao máximo as vantagens do LaTeX para apresentações é realmente o público acadêmico e institucional. Afinal, para esse público, todas as seguintes características serão interessantes: a estruturação lógica da apresentação, o foco no conteúdo e a possibilidade de criar e reaproveitar modelos de apresentação com uma mesma identidade visual.

Em todo caso, mesmo que o leitor não se enquadre nesse público ideal, se ele já estiver chegado até esse ponto do livro, isso significa que falta muito pouco para ele aprender como fazer apresentações no LaTeX. Como veremos, depois de dominar os princípios básicos do LaTeX, o esforço adicional para fazer apresentações é mínimo. Quase todos os comandos que utilizaremos nesse capítulo já foram amplamente discutido em seções anteriores desse livro. Portanto, porque não gastar apenas mais alguns minutos e aprender uma aplicação “heterodoxa” do LaTeX?

Como iremos repetir muitas noções já explicadas anteriormente, esse capítulo será consideravelmente mais curto que os demais. Como sempre fizemos ao longo do livro, irei apenas expor os fundamentos básicos sobre como gerar apresentações básicas no LaTeX. Caberá ao leitor que necessitar de algo mais avançado consultar a documentação especializada.

## Criando o documento final de uma apresentação

Gostaríamos de chamar a atenção para um detalhe importante ao gerar as apresentações. Queremos lembrar a importância para o leitor dos arquivos temporários: as apresentações *também* o utilizam bastante. Portanto, é interessante fazer com que o LaTeX gere cada apresentação ao menos *duas vezes*; exceto, claro, na utilização de referências bibliográficas, que exigirão as etapas adicionais descritas no capítulo anterior.

Em todo caso, o leitor ficará feliz em saber que sua apresentação final será transcrita no formato PDF. Este tipo de documento não é utilizado com tanta frequência para apresentações, mas funciona exatamente do mesmo modo. É possível visualizar o documento final como um texto (passando continuamente as páginas como se fosse um documento comum), mas também é possível visualizar o documento *como apresentação*, exatamente como ocorreria caso se tratasse de uma apresentação feita no OpenOffice ou no Power Point. Basta que, ao projetar o documento, o leitor procure no leitor de PDF o comando para mostrar o documento em modo de apresentação. Ao fazer isso, terá acesso a todos os recursos visuais tradicionais e poderá ir passando os slides progressivamente como em qualquer apresentação normal. Com a única diferença – que é uma enorme vantagem do LaTeX – que a formatação final do seu documento irá

permanecer totalmente estável, independentemente do computador que você tenha que utilizar para transmití-la – o que raramente é o caso quando uma apresentação feita em um editor convencional precisa ser transmitida para outro editor.

## O preâmbulo de uma apresentação

Pergunto-me se algum leitor já terá adivinhado como é possível produzir apresentações no LaTeX. Para encontrar essa resposta, basta nos voltarmos para a primeira linha dos nossos arquivos em LaTeX: para a definição da *classe* do documento. Como já foi amplamente discutido ao longo desse livro, o comando “\documentclass” indica o *tipo* de documento que estamos produzindo, afetando globalmente todas as características visuais do produto final. Por isso, caso nosso objetivo seja utilizar o LaTeX para criar uma apresentação, devemos começar por aqui: pela classe do documento.

Se o leitor não adivinhou que era aqui que estava a chave do problema, certamente adivinhará meu próximo comentário: há *diversas* classes que podem ser utilizadas para a mesma finalidade. Como sempre, em se tratando do LaTeX, várias soluções alternativas foram desenvolvidas em torno de necessidades especiais. Entretanto, ao longo deste capítulo, vamos nos concentrar exclusivamente em uma única classe: a classe “beamer”.

Essa classe é bastante antiga, estável e popular, de modo que há muitos guias e pacotes desenvolvidos especificamente para ela. Como sempre, o leitor não deve se sentir de modo algum obrigado à utilizá-la; apenas a consideramos o melhor modo de *começar* a utilizar o LaTeX para produzir apresentações. Caso tenha necessidades mais avançadas, certamente encontrará outras alternativas nos fóruns de discussão.

Ao declararmos para o LaTeX que iremos produzir um documento na classe “beamer”, podemos aproveitar e indicar outras características secundárias enquanto *opções* (na seção da variável secundária do comando) do nosso documento. Por exemplo, o seguinte comando diz ao LaTeX que iremos utilizar a classe “beamer” e que o tamanho da fonte do nosso documento será de 10pt:

```
\documentclass[10pt]{beamer}
```

O restante do preâmbulo deverá ser bastante simplificado: nem todo pacote que funciona em um documento normal irá funcionar em uma apresentação. Em todo caso, nosso velho conhecido, o pacote *xltxtra* deve aparecer novamente, para manter nossos acentos funcionando. Portanto, nossa segunda linha pode seguir a tradição dos exemplos deste livro:

```
\usepackage{xltxtra}
```

Ainda no preâmbulo, você pode definir um “tema” para a sua apresentação. O “tema” é um conceito específico dos documentos produzidos para a classe “beamer”. Sua implementação segue um comando basta simples: `\usetheme{nome-do-tema}`.

O *tema* é um *modelo visual* do documento. Ele irá dizer ao LaTeX as características visuais gerais do documento: o padrão de cores, a fonte, organização do texto na tela, etc. Embora tudo isso possa ser definido individualmente dentro de cada “slide”, é preferível incluir o máximo de definições visuais globais no preâmbulo do seu documento, pois desse modo você pode se preocupar apenas com o conteúdo enquanto estiver trabalhando em cada slides. Essa separação entre *forma* e *conteúdo*, em se tratando de apresentações no LaTeX, é feita principalmente a partir da definição do *tema* (theme) que irá afetar toda a apresentação do documento.

Assim com em relação às classes, sua distribuição do LaTeX já traz uma série de temas pré-instalados. No entanto, caso nenhum deles lhe agrade, é possível encontrar novos temas na internet. Como sempre, caso tenha interesse, o leitor pode produzir seu próprio tema. No entanto, no início, é consideravelmente mais simples utilizar alguns dos temas previamente elaborados. Assim como fizemos em relação aos estilos bibliográficos, o LaTeX irá primeiro procurar pelos novos temas na mesma pasta do arquivo utilizado e depois em sua própria biblioteca. Portanto, caso queira testar um tema, basta baixar o arquivo, deixá-lo na mesma pasta do seu documento e alterar o comando “`\usetheme`” para incluir o nome do tema. Note-se que é importante colocar precisamente o nome oficial do tema, senão o LaTeX não irá encontrá-lo; normalmente, esse nome está colocado de modo bastante evidente na documentação do tema.

Novamente, uma vantagem do LaTeX neste ponto é a independência entre o aspecto visual e o conteúdo do texto. Caso você esteja cansado do seu tema, você não precisa tocar no conteúdo da apresentação: basta mudar o comando do preâmbulo e o LaTeX irá gerar uma apresentação inteiramente nova.

Alguns temas bastante utilizados são Warsaw, Berkeley, Singapore e Frankfurt. Particularmente, gosto do tema “Amsterdam” que pode ser facilmente encontrado na internet (o leitor atento certamente terá notado que a classe *beamer* tem como convenção dar nomes de cidades aos seus temas). Além disso, cada tema possui diferentes combinações de cores, que podem ser acessíveis pelo comando: `\usecolortheme{nome do esquema de cores}`. Alguns esquemas pré-instalados: albatross, beetle, dove, beaver (o leitor atento perceberá que há outro padrão aqui também).

Recomendamos que o leitor faça diferentes experimentos com os temas e escolha seus preferidos. Se continuar insatisfeito, pode encontrar novos temas online ou mesmo criar suas próprias variações.

Ainda no preâmbulo, o autor pode adicionar informações “meta-textuais” que o *beamer* poderá utilizar de diferentes formas ao longo do seu documento. Como exemplo, eis o preâmbulo de uma apresentação básica com algumas informações adicionais:

```
\documentclass[10pt]{beamer}
\usepackage{xltextra}
\usetheme{Frankfurt}
```

```
\usecolortheme{dove}  
\title[Um subtítulo]{O título principal}  
\author{Lucas Mafaldo}  
\institute{Academia LuTeX}  
\date{\today}
```

## A estrutura de uma apresentação

Nesta seção, vamos falar daquilo que *não* muda entre um documento e uma apresentação no LaTeX. Em ambos os casos, todo o conteúdo da sua apresentação deve estar contido dentro dos comandos “`\begin{document}`” ... “`\end{document}`”. Além disso, você também deve utilizar as mesmas indicações de seções e subseções para organizar sua apresentação. Vários temas do *beamer* irão utilizar essas indicações para criar o sumário, numerar os slides e colocar várias pistas visuais para guiar o leitor ao longo da sua apresentação.

Tendo isso em mente, *antes mesmo* de preparar os slides, você já pode começar a preparar a estrutura da sua apresentação de um modo muito semelhante ao que faria caso estivesse redigindo um artigo acadêmico (podendo, inclusive, aproveitar um artigo como base para fazer a apresentação). Portanto, os seguintes códigos podem tanto compor o corpo de um artigo como o corpo de uma apresentação (como já os conhecemos bem, não precisam de novas apresentações):

```
\begin{document}  
\section{Introdução}  
Texto da introdução  
\section{Metodologia}  
Texto da metodologia  
\section{Discussão dos dados}  
\subsection{Dados encontrados}  
Conteúdo \emph{desta} subseção  
\subsection{Análise dos dados}  
Conteúdo desta subseção  
\section{Considerações finais}  
Conteúdo desta seção\footnote{Esta é a última seção}.  
\end{document}
```

## O corpo da apresentação

A grande diferença entre utilizar o LaTeX para gerar uma apresentação e um texto está no modo com as páginas são utilizadas. Vimos que, ao produzir um texto no LaTeX, não precisamos nunca nos preocupar com o que vai aparecer em cada página: o LaTeX automaticamente calcula as quebras de página, o espaço para as notas de rodapé e mesmo a divisão silábica. No entanto, em se tratando de uma apresentação, isso evidentemente não faz sentido: uma característica essencial desse tipo de documento é a necessidade de sabermos exatamente o conteúdo de cada página.

Por isso, ao produzirmos apresentações no LaTeX, as páginas não são geradas automaticamente: existe um código para determinar o início e o fim de cada página da apresentação. Esse código é bem parecido com vários que já vimos, pois também se trata da criação de um “ambiente”. No caso, o leitor deve utilizar o código “`\begin{frame}`” para indicar que o slide começou e código `\end{frame}` para indicar que ele chegou ao fim.

Tudo que estiver *dentro* desse ambiente irá aparecer em uma única página da apresentação. Dentro desse ambiente, no entanto, o LaTeX volta a calcular automaticamente a distribuição de espaço *a partir* das características globais do documento definidos no preâmbulo (tema, fonte, etc.).

Novamente, o LaTeX é bastante inteligente para calcular essa distribuição do espaço e tentará sempre respeitar as regras gerais estabelecidas pelo preâmbulo. Entretanto, como seu espaço é consideravelmente mais limitado do que seria em relação a um texto normal (ele não pode simplesmente iniciar um nova página como faria em um artigo, por exemplo), o LaTeX não será capaz de fazer nenhuma mágica caso você coloque conteúdo demais em cada página da apresentação. Por isso, enquanto nós podemos ficar totalmente tranquilos em relação ao modo como o LaTeX organiza nossos textos, o mesmo grau de tranquilidade simplesmente não é possível em relação às apresentações: depois de gerar o nosso documento, é importante verificar se a aparência de cada slide está aceitável.

Via de regra, no entanto, desde que o leitor se lembre do princípio de não colocar conteúdo demais em cada página – inclusive, uma recomendação amplamente aceita por princípios didáticos – o LaTeX irá organizar seu texto de um modo bastante eficiente e adequado às exigências do tema escolhido por você.

Note-se ainda que, por definição, todo o conteúdo que está dentro de um slide aparecerá na tela ao mesmo tempo. No entanto, há inúmeros comandos que podem modificar essa aparência. Como nosso foco não é fazer uma apresentação exaustiva dos efeitos visuais, vamos apenas fornecer um comando como exemplo: o “`\pause`” faz que a apresentação realiza uma “pausa” no local onde o comando foi adicionado, requerendo que o palestrante aperte o comando para o conteúdo do restante da página aparecer. Comandos semelhantes e alternativos podem ser facilmente encontrado em guias na internet.

Por fim, veremos um último comando específico à classe *beamer*: o comando “`\titlepage`”. O comando faz exatamente aquilo que seu nome indica: ele cria uma página-título com todos os “meta-dados” sobre o documento presentes no preâmbulo (autor, data, título, instituição, etc.).

Em suma, vamos revisar as principais diferenças entre o corpo de um *texto* e uma *apresentação* no LaTeX:

1. Cada página da apresentação é um “ambiente” cujo início e fim devem ser identificando

- com os comandos “`\begin{frame}`” e “`\end{frame}`”.
2. O LaTeX irá automaticamente calcular como organizar todo o conteúdo que estiver entre esses dois comandos em um única página – o que significa: tenha o cuidado em não colocar conteúdo demais!
  3. Todo o conteúdo de um página irá aparecer ao mesmo tempo se o autor não utilizar comandos especiais para criar exceções na apresentação visual.
  4. Um exemplo de comandos deste tipo é o comando “`\pause`”, que cria pequenos intervalos na apresentação, exatamente no ponto do texto onde eles estão inseridos.
  5. O comando “`\titlepage`” cria uma página de título com os dados sobre o documento presentes no preâmbulo. Ele deve ser colocado dentro de uma página, isto é, entre os comandos “`\begin{frame}`” e “`\end{frame}`”.

Além desses comandos, o leitor pode utilizar a maioria dos comandos padrões para o LaTeX, como vimos no código exemplificado na seção anterior.

## O modelo de uma apresentação

Nas três seções anteriores, o leitor aprendeu os três pontos seguintes: 1. Os comandos para colocar a classe *beamer* no preâmbulo; 2. Os comandos normais do LaTeX que também funcionam nesta classe; 3. Os comandos específicos ao *beamer* para criar uma apresentação básica.

Que tal agora misturar tudo e criar uma apresentação básica? O leitor certamente já sabe o suficiente para fazer seus próprios experimentos. Em todo caso, o código abaixo, retirado do arquivo “`lutex18.tex`”, irá fornecer um exemplo de como podemos integrar os comandos do LaTeX que já conhecemos com os comandos específicos ao *beamer* que vimos nas seções anteriores.

Sugerimos que o leitor tente ler o código-fonte inteiro seguinte e verifique se comprehende o que cada comando está fazendo. Em caso de dúvida, basta voltar às seções anteriores e encontrará explicações para cada um deles

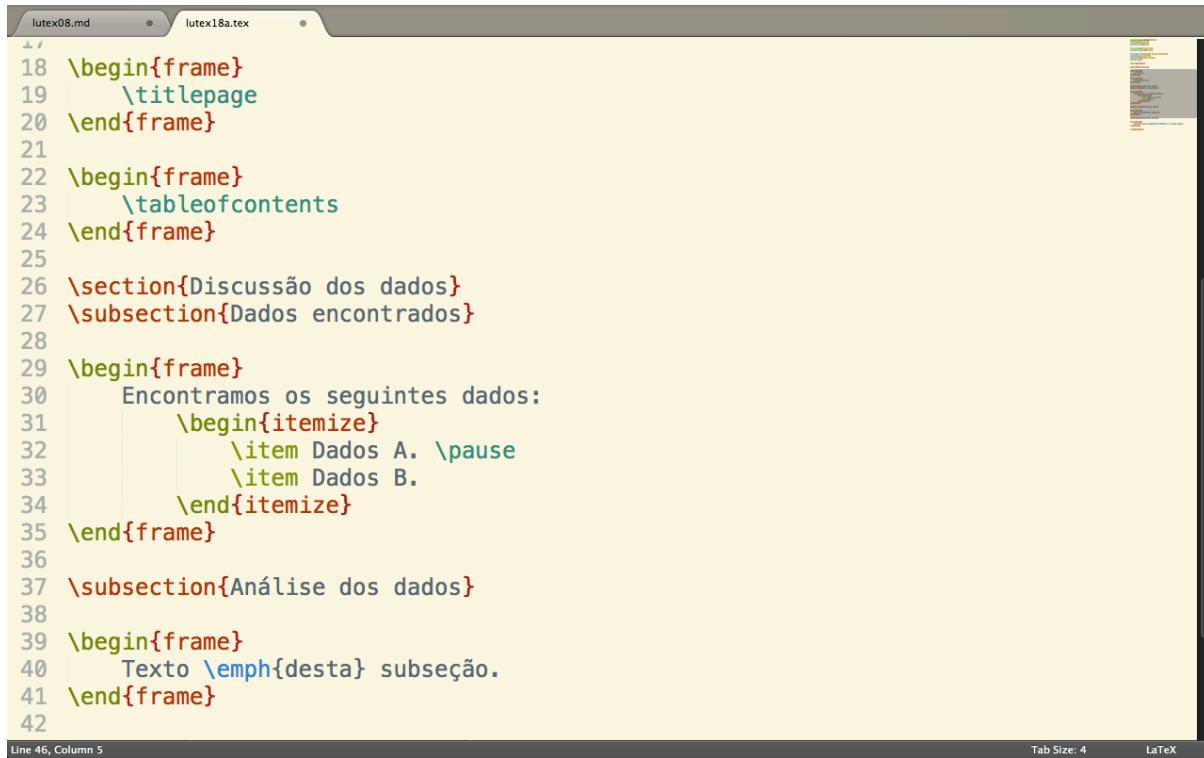
```
\documentclass[10pt]{beamer}
\usepackage{xltextra}
\usetheme{Frankfurt}
\usecolortheme{dove}
\title[Um subtítulo]{O título principal}
\author{Lucas Mafaldo}
\institute{Academia LuTeX}
\date{\today}
\begin{document}
\section{Introdução}
\begin{frame}
```

```
\titlepage
\end{frame}
\begin{frame}
\tableofcontents
\end{frame}
\section{Discussão dos dados}
\subsection{Dados encontrados}
\begin{frame}
Encontramos os seguintes dados:
\begin{itemize}
\item Dados A. \pause
\item Dados B.
\end{itemize}
\end{frame}
\subsection{Análise dos dados}
\begin{frame}
Texto \emph{desta} subseção.
\end{frame}
\section{Considerações finais}
\begin{frame}
Conteúdo desta seção\footnote{Esta é a última seção}.
\end{frame}
\end{document}
```

Todos os elementos desta seção são bem conhecidos do leitor, mas gostaríamos de chamar atenção a algo que podemos fazer *durante* a redação do código que *não* irá afetar o documento final. Como é bastante confuso lidar com vários códigos parecidos (inúmeros “`\begin{isso}`”, “`\end{aquilo}`”, etc.), nós ocasionalmente adicionamos espaços adicionais *antes* do início das linhas de comandos, para ficar mais fácil *para nós* visualizar onde cada seção começa e termina. Como vimos no início do livro, o LaTeX trata os espaços de modo lógico e não quantitativo. Portanto, para ele não faz diferença *quanto* espaço colocamos entre os comandos. Por causa disso, em se tratando de um código mais complexo, optamos por utilizar esses passos para tornar a leitura do código mais fácil. Novamente, repetimos que é um procedimento principalmente utilizado para facilitar a leitura do código-fonte sem qualquer efeito sobre o texto final – e, novamente, é algo que os programadores fazem com frequência, mas que parecerá estranho a quem vem das humanidades.

Para entender do que estamos falando, observem como parte desse longo código aparece no

Sublime Text:



```

18 \begin{frame}
19   \titlepage
20 \end{frame}
21
22 \begin{frame}
23   \tableofcontents
24 \end{frame}
25
26 \section{Discussão dos dados}
27 \subsection{Dados encontrados}
28
29 \begin{frame}
30   Encontramos os seguintes dados:
31   \begin{itemize}
32     \item Dados A. \pause
33     \item Dados B.
34   \end{itemize}
35 \end{frame}
36
37 \subsection{Análise dos dados}
38
39 \begin{frame}
40   Texto \emph{desta} subseção.
41 \end{frame}
42

```

Imagen do código-fonte do arquivo “lutex18a.tex”

Embora o arquivo seja bem longo do que os demais que vimos ao longo do livro, a distância entre os espaços e a diversidade de cores torna consideravelmente fácil navegar entre o texto e os códigos. Por isso, reforçamos nossa recomendação de que o leitor procure um editor que se adapte ao seu estilo de trabalho.

Por fim, é interessante notar como esse documento incorpora quase os elementos vistos anteriormente. Por isso, prestar atenção ao seu código fonte equivale a fazer uma revisão geral do livro.

## Analisando o resultado final

Para demonstrar os recursos de uma apresentação preparada pelo *beamer* e a versatilidade dessa classe, nós faremos o seguinte nesta seção: vamos inserir imagens comparando exatamente o mesmo exemplo alterando apenas o tema utilizado para ver como o documento é afetado.

O primeiro documento será gerado exatamente pelo código exposto anteriormente (retirado do arquivo “lutex18a.tex”), enquanto a segunda imagem utilizará um tema que precisa ser retirado da internet (o tema “Amsterdam”, utilizando no arquivo “lutex18b”).

Observem como há uma série de recursos interessantes que são automaticamente inseridos pelo *beamer*. Esses recursos nos remetem a um dos princípios do LaTeX: embora exija mais esforço desenvolver um primeiro documento bem estruturado, esse esforço é amplamente recompensado pelo acesso à várias funcionalidades e um documento final consideravelmente mais rico, estável e flexível.

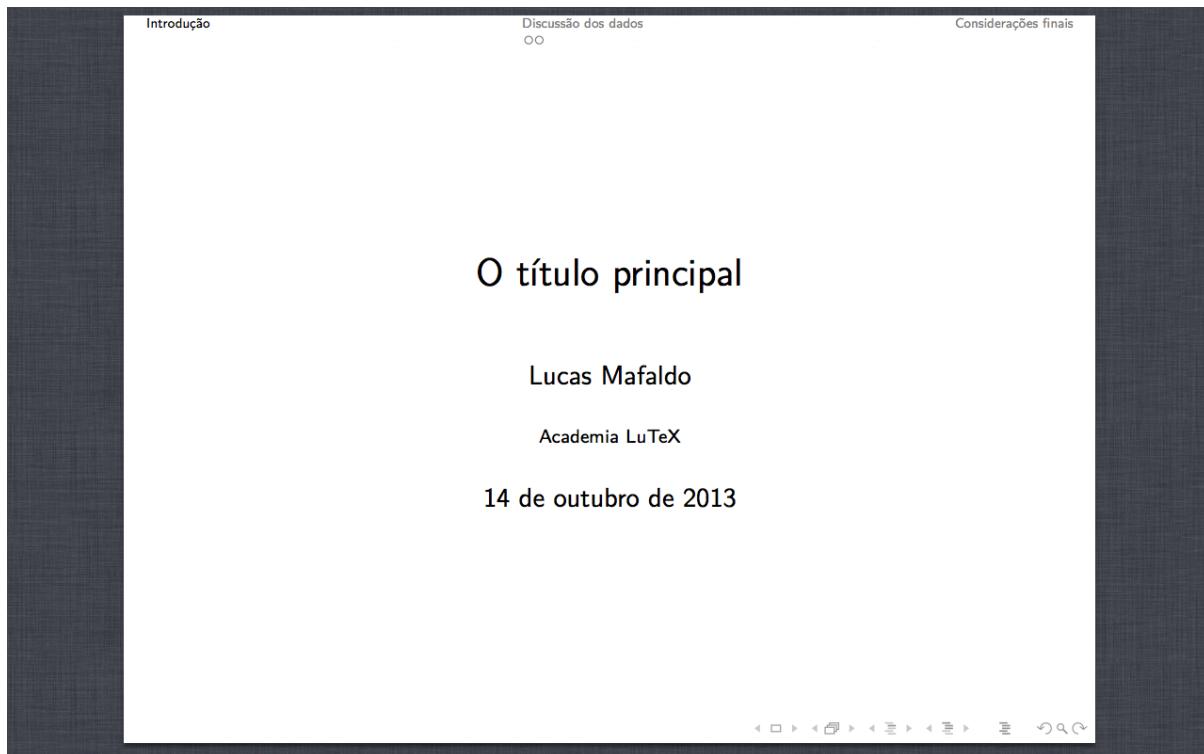


Imagen da primeira página do arquivo “lutex18a.pdf”

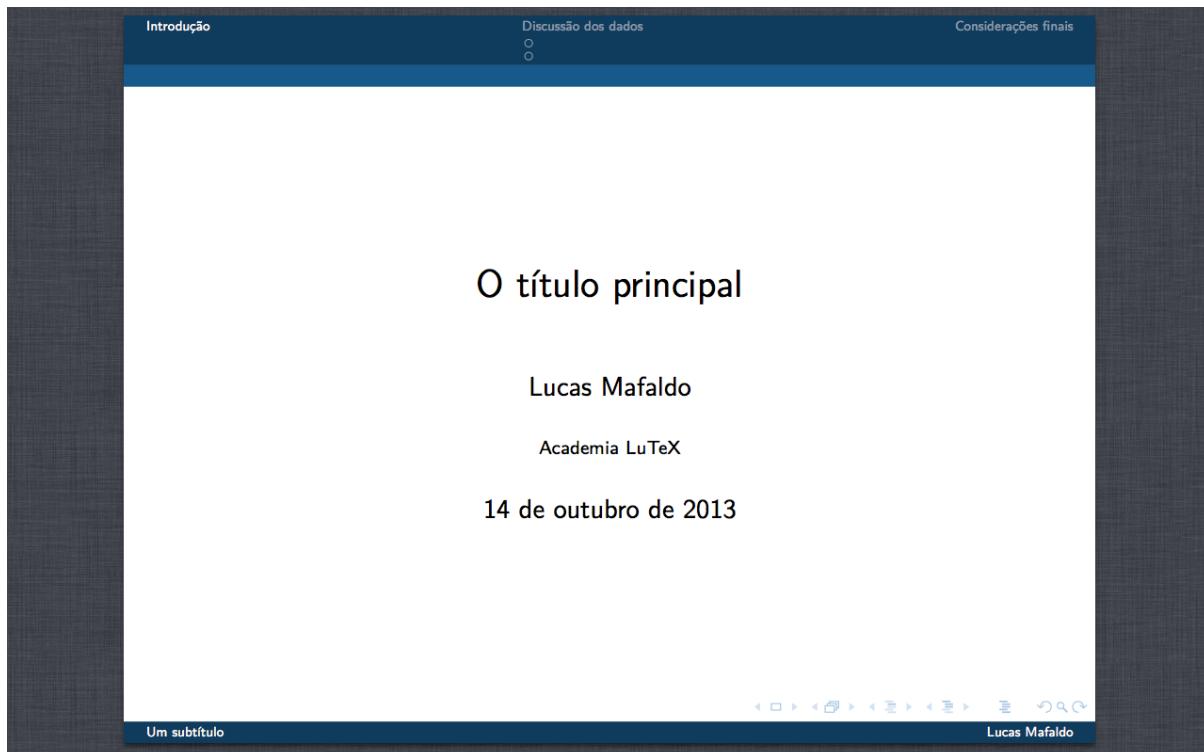


Imagen da primeira página do arquivo “lutex18b.pdf”

O que podemos notar nessa primeira página, gerada automaticamente pelo comando “\titlepage”, é que há uma série de recursos visuais também inseridos de modo automático pelo LaTeX. Se nos voltarmos para o topo da página, veremos que as três seções (“Introdução”, “Discussão” e

“Considerações finais”) são apresentadas ao leitor. Note-se também que a seção “Introdução” está em uma fonte diferente para frisar que estamos nesta seção em ambos os modelos, embora no tema “Amsterdam” essa diferença fique mais clara.

O modelo “Amsterdam” traz uma vantagem em relação ao modelo “Frankfurt”: ele utiliza o rodapé da página para transmitir ao público informações adicionais: no caso, ao longo de toda a apresentação, o autor pode determinar que um subtítulo fique justificado à esquerda, enquanto o próprio nome fica justificado à direita. Esse tipo de formatação é relevante quando o objetivo da apresentação está atrelado ao reforço de um marca ou identidade específica. Note-se que, dependendo do tema utilizado, é possível utilizar também o rodapé para inserir o número de slides da apresentação e apontar para o progresso da palestra.

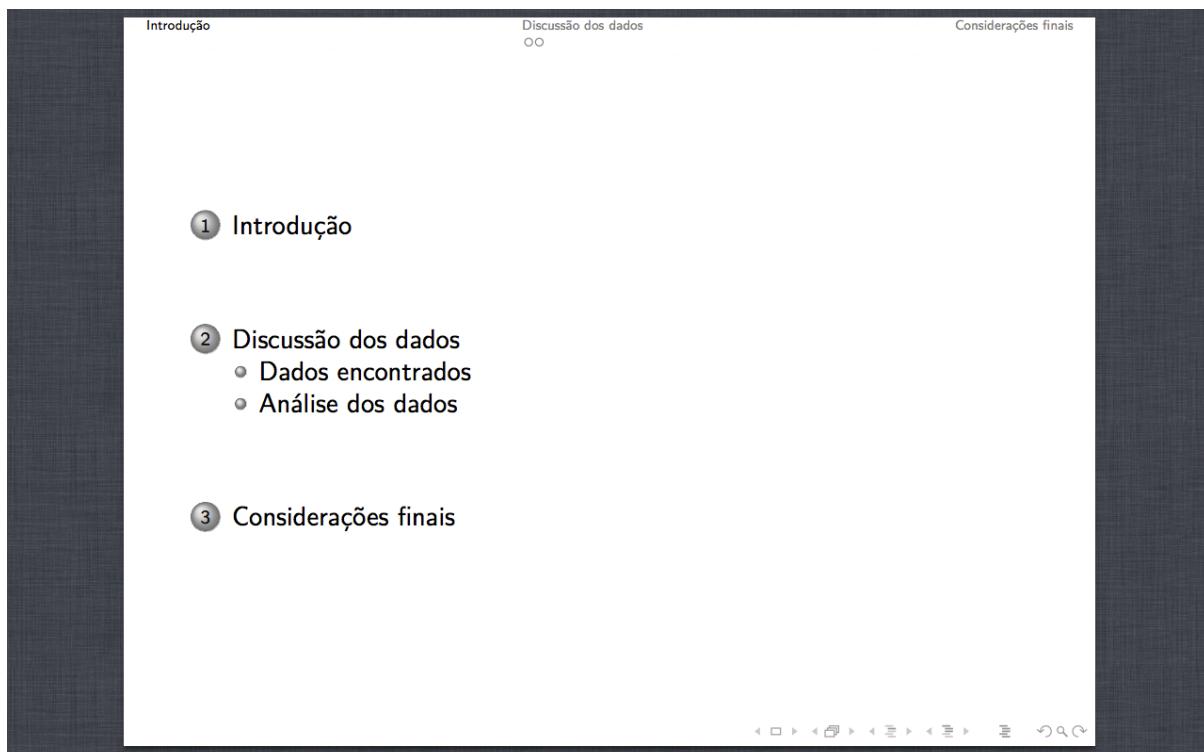


Imagen da segunda página do arquivo “lutex18a.pdf”

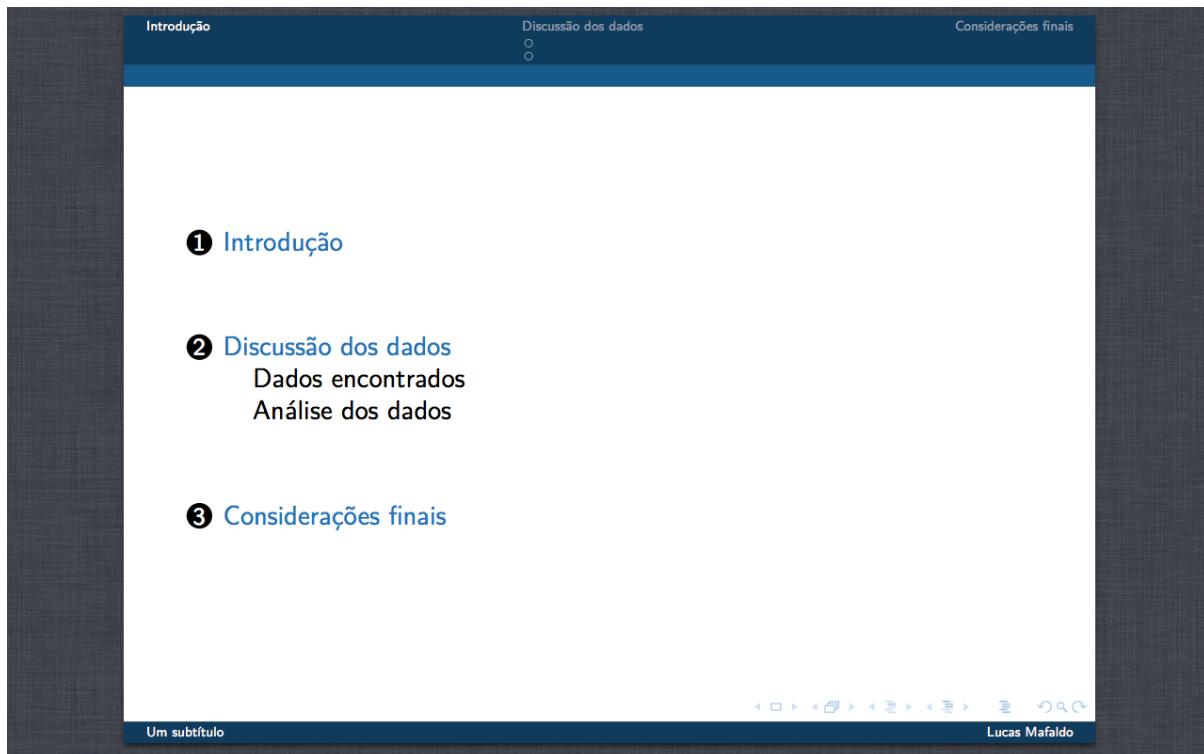


Imagen da segunda página do arquivo “lutex18b.pdf”

Caso o leitor acompanhe essas imagens observando o código-fonte descrito acima, verá que a segunda página da nossa apresentação correspondem ao segundo ambiente *frame* criado. O conteúdo desse ambiente consistia apenas no código “\tableofcontents” que criou automaticamente o sumário da nossa apresentação.

O tema Amsterdam continua fazendo uso do rodapé da apresentação, como o fará ao longo de todo o documento. Tirando isso, não há novas diferenças estruturais entre os dois temas nessa página; ambos os temas diferem apenas na aparência visual.

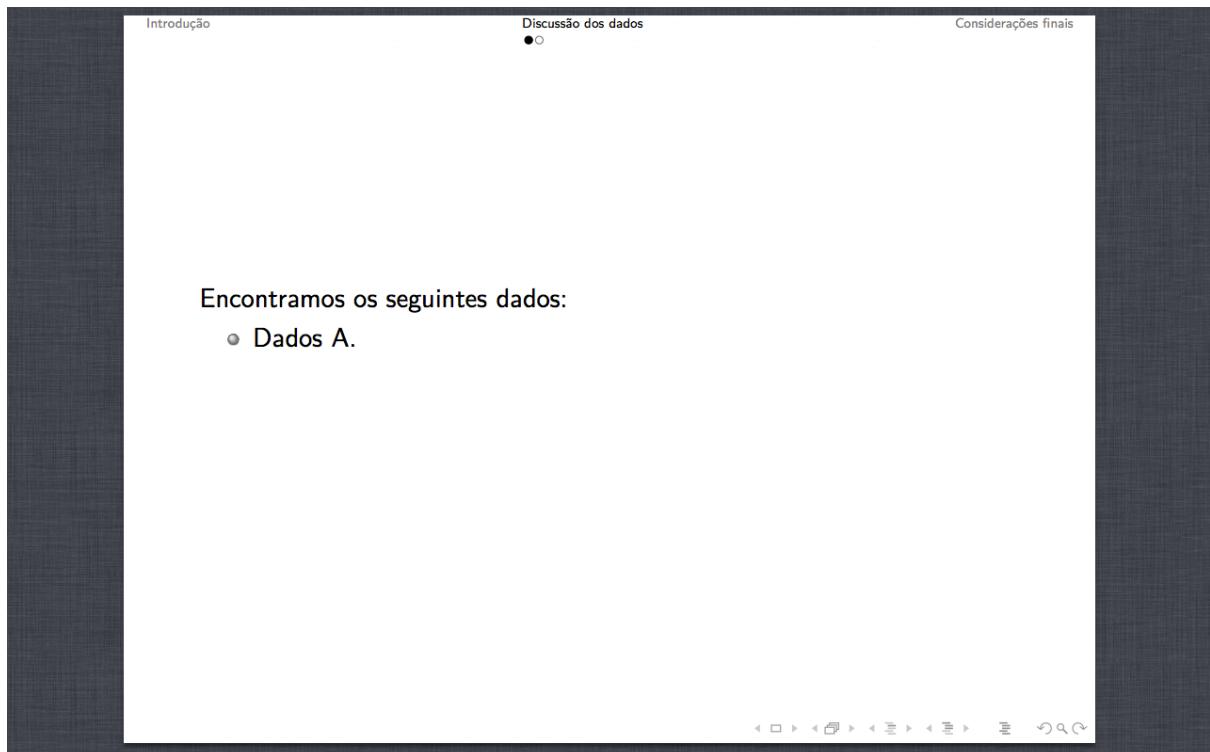


Imagen da terceira página do arquivo “lutex18a.pdf”

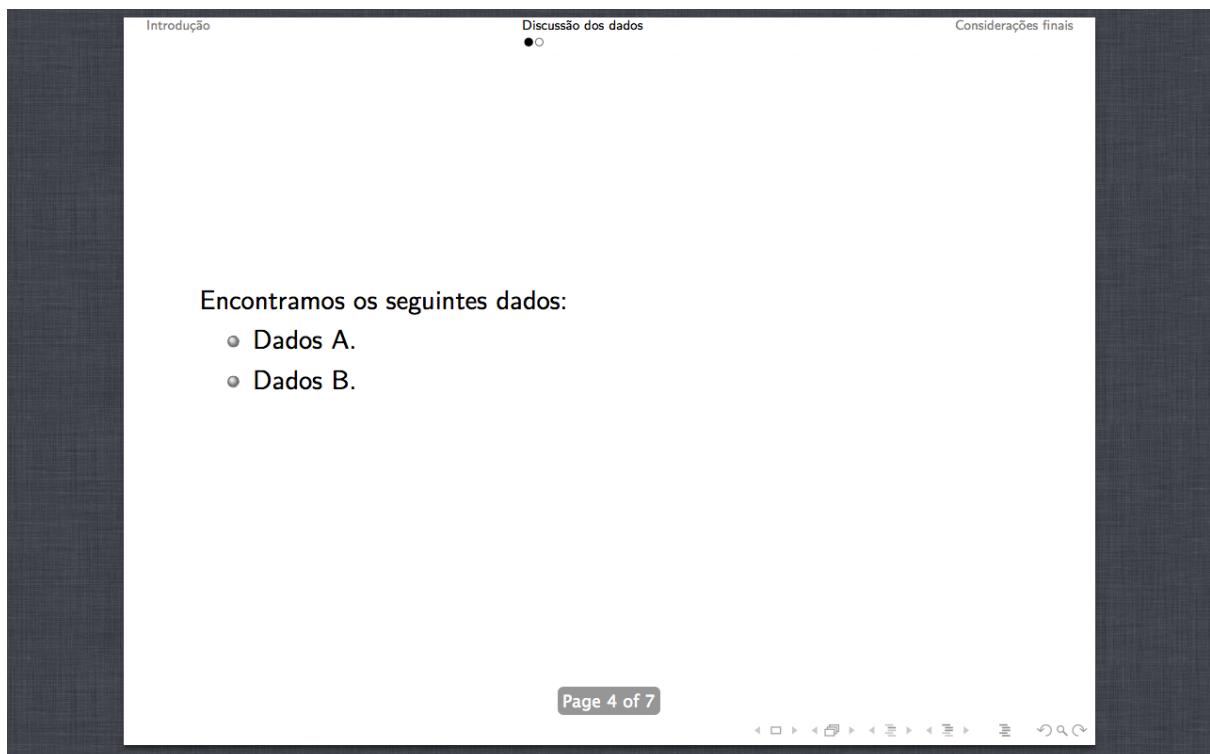


Imagen da terceira página do arquivo “lutex18a.pdf”

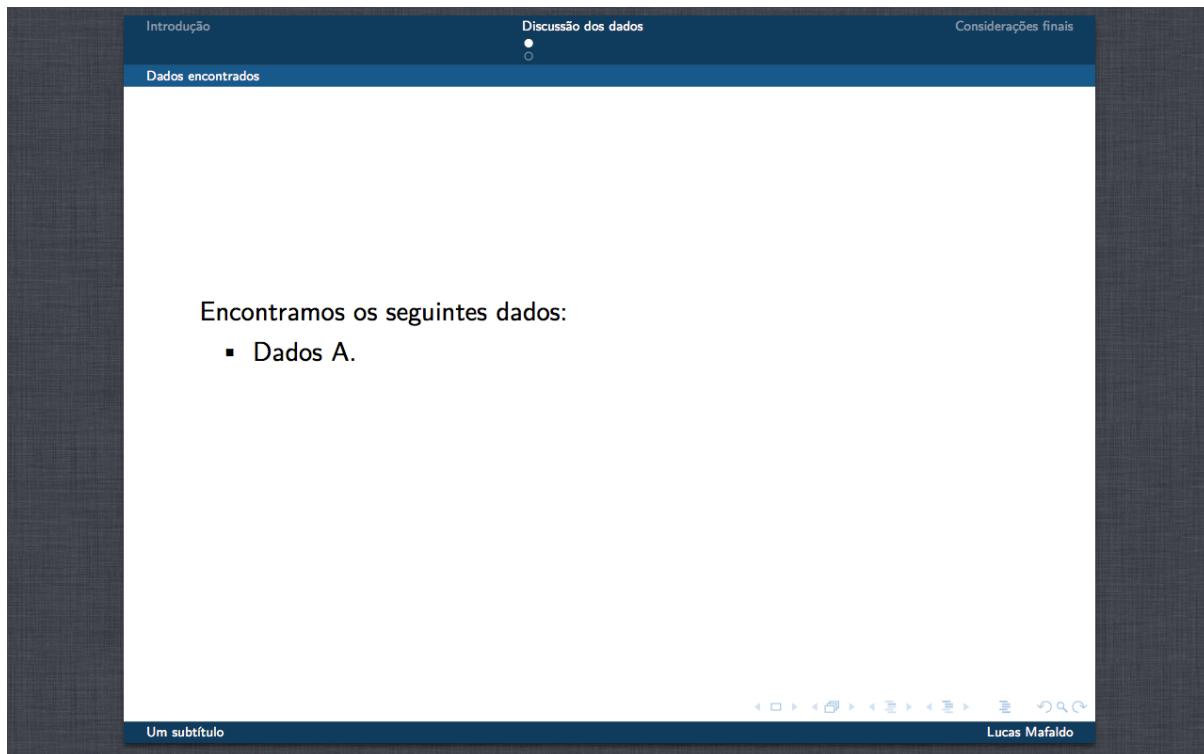


Imagen da terceira página do arquivo “lutex18b.pdf”

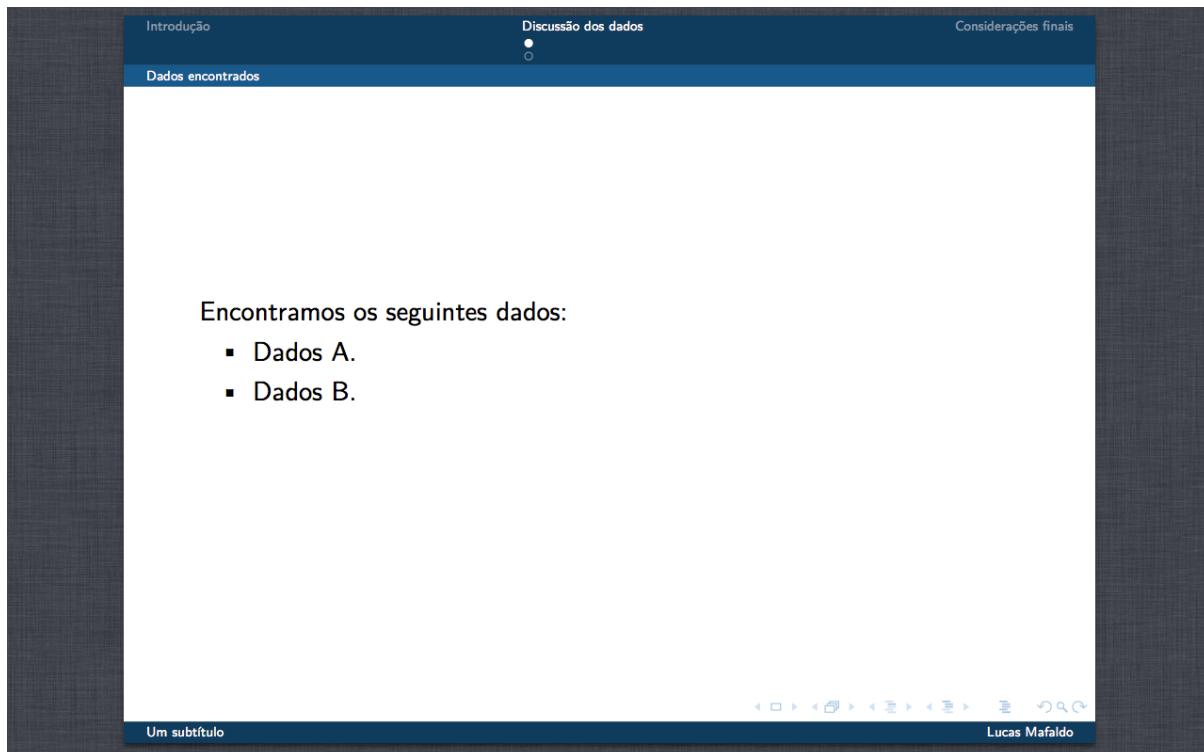


Imagen da terceira página do arquivo “lutex18b.pdf”

Notem que a terceira página aparece em duas vezes. Voltando ao código, encontraremos o motivo disso: o comando “`\pause`” cria um intervalo exatamente no ponto da página em que ele aparece. Pode-se utilizar esse comando repetidas vezes para criar quantos intervalos forem

necessários. Além desses comandos específicos à classe *beamer*, se o leitor voltar ao código-fonte, verá que fizemos uso do ambiente para gerar listas – mostrando que podemos reutilizar nosso conhecimento dos capítulos anteriores também em nossas apresentações.

Notem também como o topo da página está sendo modificado na medida em que a apresentação avança: na parte superior, em *ambos* os temas, temos o título da seção em que estamos. Na medida em que mudamos de seção, a fonte muda para dar mais ênfase à seção onde estamos (novamente, isso fica mais claro no tema Amsterdam, apesar de aparecer em ambos os casos).

Além disso, percebam que há um novo elemento de visualização: pequenos círculos indicam em que ponto da subseção da seção maior estamos (eles não aparecem nas outras seções, porque apenas esta possui subseções). Novamente, esse critério de navegação é criado pelo LaTeX automaticamente a partir dos comandos que estruturam o nosso texto. Por fim, notem ainda um recurso que aparece apenas no tema Amsterdam: o título da subseção aparece entre o cabeçalho e o texto (“Dados encontrados”).



Imagen da quarta página do arquivo “lutex18a.pdf”

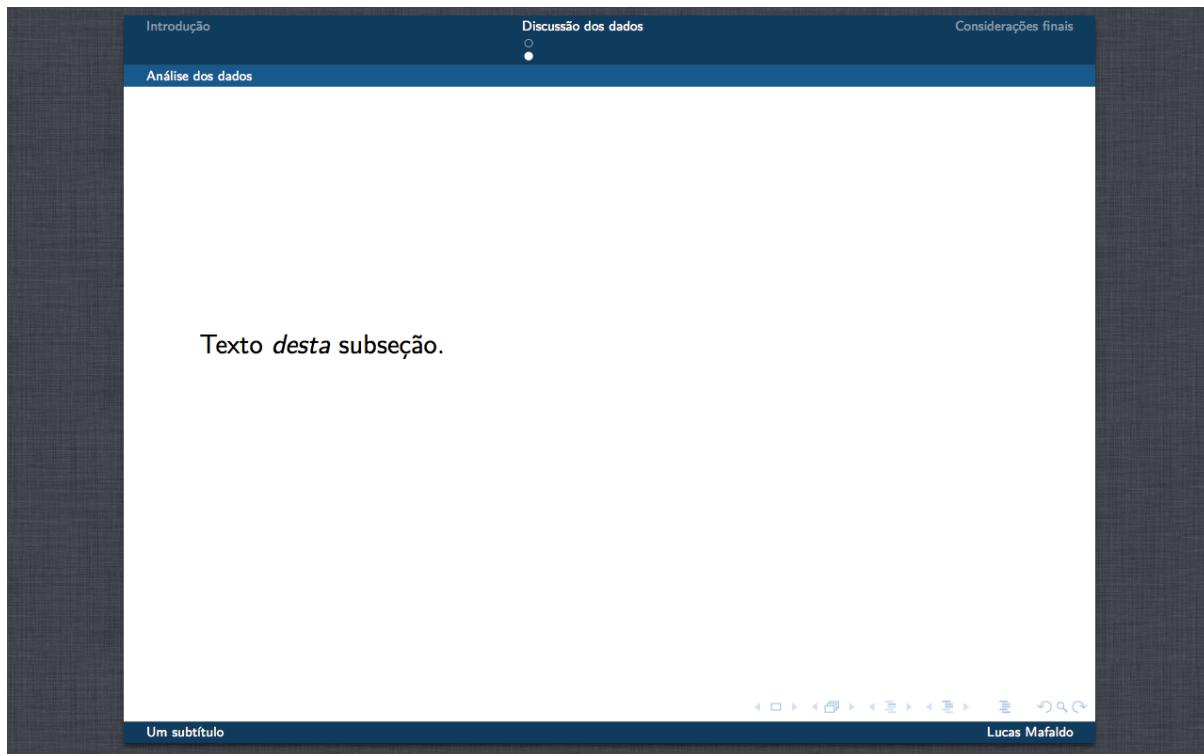


Imagen da quarta página do arquivo “lutex18b.pdf”

Nesta seção, não temos elementos visuais inteiramente novos, mas podemos observar como cada tema continua seguindo o padrão previamente estabelecido. Em ambos os casos, o círculo no topo da página mudou de cor para assinalar a mudança da subseção. No caso do tema Amsterdam, essa mudança também foi assinalada pela inserção do novo subtítulo (“Análise dos dados”). Note-se ainda que o trecho em itálico serve para nos lembrar que os comandos normais do LaTeX ocasionalmente podem ser reutilizados nesta classe.

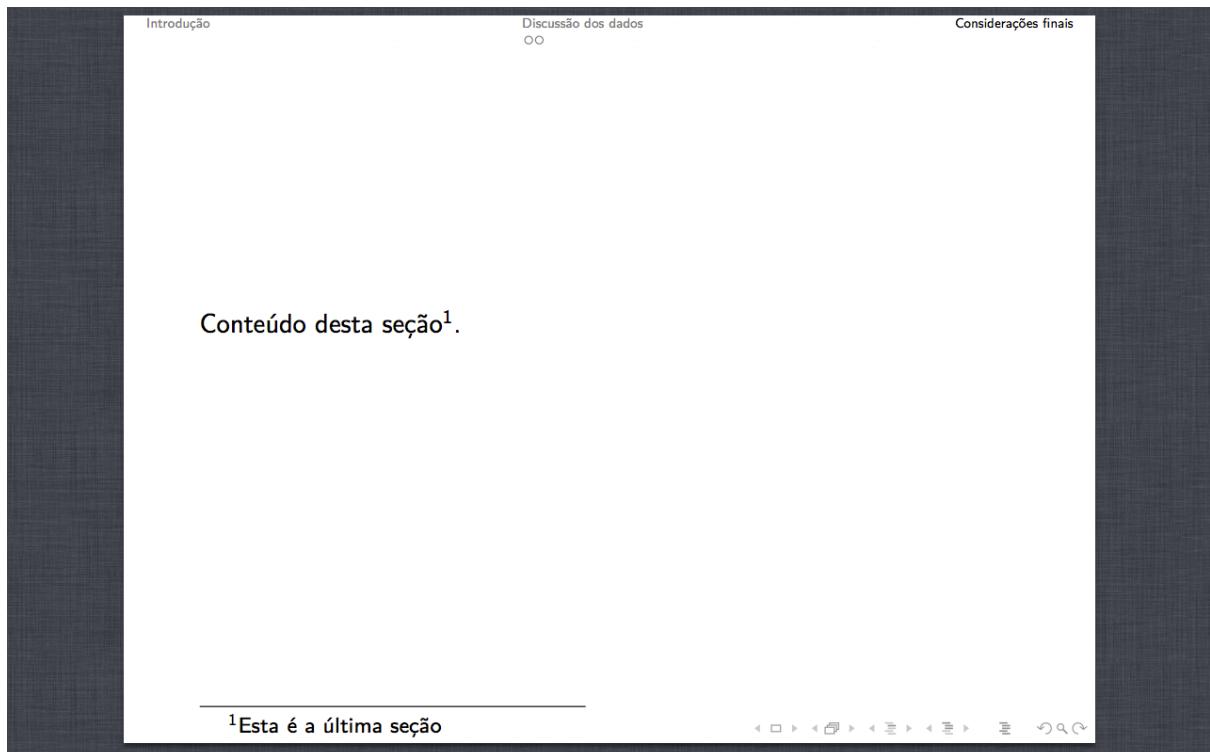


Imagen da quinta página do arquivo “lutex18a.pdf”

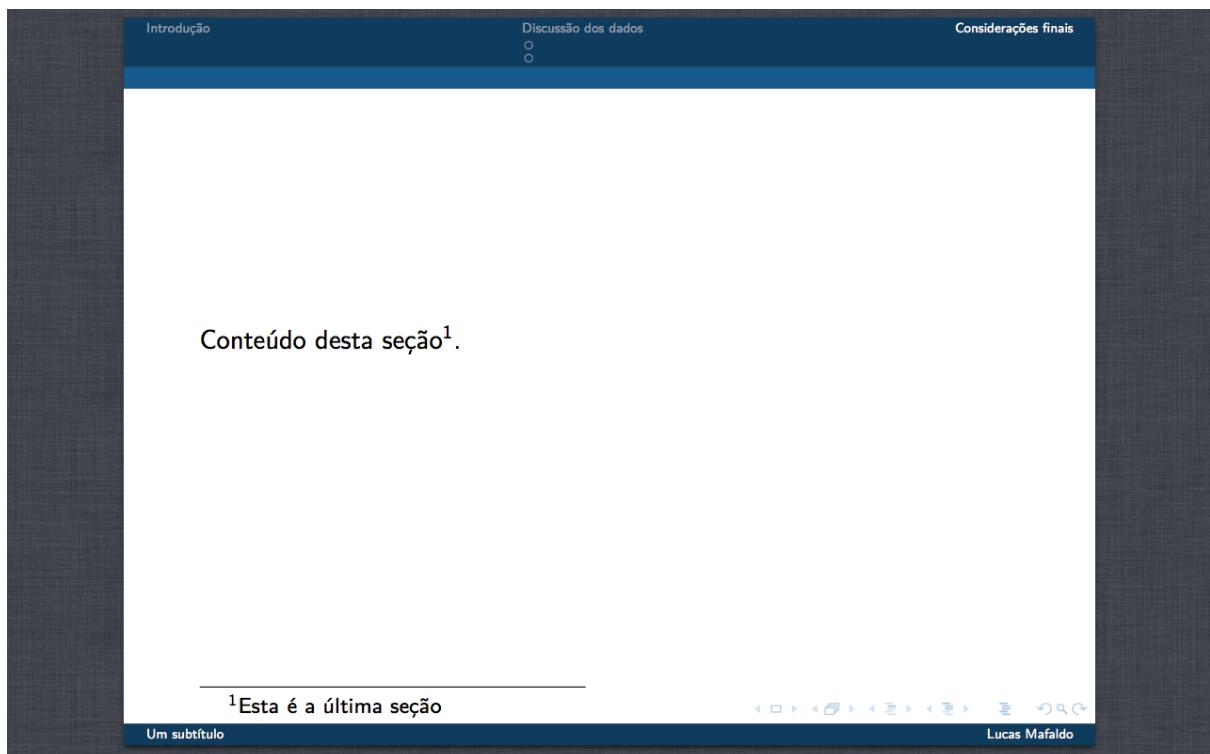


Imagen da quinta página do arquivo “lutex18b.pdf”

Por fim, chegamos ao final do nosso documento, o que está assinalado no topo do documento, quando a ênfase da fonte passa para a última categoria. O leitor também tem a oportunidade de voltar ao código-fonte e revisar o comando para a criação de notas de rodapé: como pode-

se ver, exatamente o mesmo comando poder ser utilizado também nessa classe, pois o LaTeX automaticamente calcula o melhor modo de transcrevê-lo em um espaço visual diferente.

Novamente, caso o leitor não considere essas apresentações visualmente atraentes, lembre-se que é bastante fácil mudar a combinação de cores e encontrar novos temas na internet. Além disso, é até mesmo possível produzir seus temas inteiramente novos.

## Notas finais

Espero ter conseguido mostrar que, tomando como base o conhecimento dos capítulos anteriores, são necessários apenas alguns poucos comandos adicionais para criar uma apresentação no LaTeX.

Os princípios básicos do LaTeX são os mesmo em ambos os casos. Além disso, há vários comandos que podem ser reaproveitados. A única diferença é a necessidade de adicionar alguns poucos comandos específicos para organizar as páginas da sua apresentação.

Em relação ao conhecimento específico para as apresentações, esses são os pontos essenciais:

1. Mudar a classe para do documento para *beamer* com o comando “\documentclass{beamer}”.
2. Escolher o tema que irá definir as características visuais do documento com o comando “\usetheme{nome do tema}”.
3. Criar um “ambiente” chamado de *frame* para cada página da apresentação, colocando o conteúdo de cada página entre os códigos “\begin{frame}” e “\end{frame}”.
4. Lembrar-se de rodar o LaTeX ao menos duas vezes para garantir que a formatação final esteja correta (ou mais, caso utilize o BibTeX para gerar a bibliografia, o que foi explicado no capítulo anterior).

Ao final disto, o leitor terá uma apresentação no formato PDF com todas as vantagens do LaTeX. Você terá a garantia de que sua apresentação terá uma aparência estável em diferentes computadores, terá mais controle sobre a aparência final e poderá criar e editar novos documentos rapidamente, repetindo ou alterando o preâmbulo. Em suma, com apenas algumas pequenas mudanças em seu código-fonte, o leitor pode aproveitar todo o conhecimento adquirido nos capítulos anteriores e aplicar os benefícios do LaTeX – a estabilidade, o controle, a flexibilidade – também para as suas apresentações.

# Considerações finais

Ao terminar esse livro, espero ter transmitido ao leitor as ferramentas para começar a utilizar o LaTeX em seu trabalho. Mais do que transmitir informações básicas, no entanto, espero tê-lo convencido das vantagens do LaTeX e tê-lo transmitido o desejo de continuar aprendendo mais sobre esse sistema.

Queria aproveitar essas últimas páginas para enfatizar as vantagens colaborativas do LaTeX. O LaTeX é uma ferramenta extremamente flexível e expansível. Sempre existem novos pacotes e novos comandos. Por isso, ele é um sistema que cresce na medida em que mais pessoas o adotam. Embora o LaTeX seja consideravelmente vantajoso para o pesquisador isolado, seus benefícios se multiplicam quando todo um grupo o adota em conjunto.

O leitor certamente sabe que uma das dificuldade envolvidas no trabalho coletivo em torno de diversos tipos de documentos é justamente a dificuldade de manter a estabilidade do formato enquanto passa por diferentes computadores e sistemas operacionais. Devido a sua abertura, o LaTeX pode ser um código de comunicação comum em vários projetos.

Note-se também que sua separação entre conteúdo e aparência é ideal para uma série de projetos institucionais que seguem regras específicas: relatórios, avaliação dos alunos, publicações acadêmicas, enfim todo o tipo de documentação envolvida no setor de pesquisas contém uma série de regras para a sua apresentação final. Essas regras podem ser codificadas uma única vez em um padrão do LaTeX e ser automaticamente reimplantada universalmente na instituição.

Um dos motivos que me levaram a escrever esse livro foi justamente a esperança que possa haver uma adoção mais ampla do LaTeX pelos pesquisadores das humanidades. Creio que esse sistema pode nos poupar coletivamente bastante tempo e dor de cabeça ao lidar com documentos oriundos de sistemas diferentes e destinado à publicação de acordo com normas distintas.

Se o leitor estiver convencido dessas vantagens, espero que esse livro seja apenas o primeiro passo na adoção do LaTeX em sua rotina de trabalho. Espero também que esse livro já tenha todos os princípios necessários para que o leitor possa criar um modelo básico e começar a adotá-lo imediatamente.

Dito isso, reconheço que há muito conteúdo que não foi discutido. Como repetimos exaustivamente ao longo do livro, há muito mais informação sobre o LaTeX do que poderíamos indicar em uma publicação introdutória. Por isso, recomendamos que o leitor utilize os princípios adquiridos aqui para caminhar em direção a textos mais avançados e especializados.

Decidi não incluir nenhuma indicação bibliográfica específica ou recomendar nenhuma fonte em particular para não cometer nenhuma injustiça ou imprecisão. O conhecimento nesse setor, por sua própria natureza, está consideravelmente difundido em inúmeros guias, cursos, livros e documentações específicas. Mais importante do que se concentrar em uma dessas fontes em particular é que o leitor desenvolva seu faro para encontrar informações relevantes para as suas necessidades. Espero que os princípios discutidos ao longo desse livro ajudem o leitor a desenvolver essa habilidade.

Dito isso, segue em anexo o código-fonte de todos os arquivos utilizados ao longo desse livro. Espero que ajudem o leitor a criar seus próprios modelos e sirvam de empurrão inicial para trabalhar com o LaTeX.

Em uma nota final, fique registrado que o autor encontra-se inteiramente aberto às críticas e sugestões dos leitores. Decidi publicar esse livro como *ebook* justamente para colocar o mais rápido possível uma versão imperfeita diante dos leitores, para que pudesse aprimorá-lo a partir dos comentários recebidos. Pareceu-me melhor publicar o que tinha à mão – com as eventuais falhas, imprecisões e limitações – do que aguardar por uma perfeição que nunca viria.

Se o leitor chegou até aqui, agradeço sinceramente a atenção e espero que considere que seu tempo foi bem aplicado.

Lucas Mafaldo Oliveira,

Natal, 14 de Outubro de 2013.

# ANEXOS

Nesta seção, serão publicados o código-fonte de todos os exemplos utilizados ao longo deste livro. Eles aparecem exatamente na ordem em que foram utilizados ao longo do livro e, ocasionalmente, trazem alguns comentários adicionais que explicam o código.

Já que esse livro será prioritariamente distribuído no formato PDF, a solução em imprimir diretamente o código-fonte me pareceu superior ao trabalho em manter um diretório virtual com os arquivos para downloads. Desse modo, o leitor tem acesso tanto ao conteúdo do livro como a todos os exemplos em único arquivo.

Para utilizar os arquivos transcritos a seguir, o leitor precisa prestar apenas nas seguintes convenções: o título de cada seção correspondem ao nome do arquivo que originou o código e o conteúdo de cada seção traz exclusivamente o código-fonte do arquivo em questão. Caso o leitor queira utilizar esses arquivos, terá apenas que copiar o código fonte para seu editor e salvar uma cópia local com o título indicado.

Evidentemente, o leitor tem a liberdade de renomear os arquivos como lhe parecer melhor. Entretanto, ao fazer isso, é preciso se lembrar que alguns arquivos fazem referência aos outros, de modo que, caso o nome de um arquivo seja alterado, é preciso também alterar as referências feitas a ele.

Esperamos que os exemplos que seguem sejam suficientes para o leitor construir seus problemas modelos e começar a utilizar o LaTeX em sua rotina de trabalho.

## lutex01.tex

```
\documentclass{article} % Elemento necessário do preâmbulo que indica a utilização da classe "article".  
\usepackage{xltextra} % Elemento opcional do preâmbulo que carrega o pacote "xltextra".  
\begin{document} % Avisa que o "ambiente" document começou. Seu texto começa a partir dessa linha.  
O \emph{conteúdo} do seu documento fica aqui. % O comando "\emph" enfatiza as palavras dentro das chaves.  
\end{document} % Avisa que o conteúdo do seu documento terminou. É importante para que o LaTeX funcione normalmente.
```

## lutex02.tex

```
\documentclass{article}
```

```
\usepackage{xltextra}
```

```
\begin{document}
```

Esse documento tem apenas uma única linha em termos lógico. No entanto, ela é tão longa que termina por forçar o editor a dividir visualmente o texto em quatro linhas.

```
\end{document}
```

## **lutex03.tex**

```
\documentclass{article}
```

```
\usepackage{xltextra}
```

```
\begin{document}
```

Uma linha com um única frase.

Outra linha com uma única frase.

A linha anterior ficou em branco!

```
\end{document}
```

## **lutex04.tex**

```
\documentclass{article}
```

```
\usepackage{xltextra}
```

```
\begin{document}
```

Nessa primeira linha, colocaremos várias frases. No entanto, como não iremos assinalar ao LaTeX nenhuma quebra de linha, todas elas ficarão no mesmo parágrafo.

Para assinalarmos que estamos em um novo parágrafo, deixamos a linha anterior em branco. Novamente: não precisamos nos preocupar com o \emph{espaço} que esse texto ocupa em nosso editor. Precisamos apenas olhar para a numeração das linhas no arquivo original em LaTeX e verificar que cada parágrafo está em uma única linha

Nesse terceiro parágrafo, faremos algo diferente.

Vamos colocar cada frase em linha separada.

No entanto, como não deixamos nenhuma linha em branco, o LaTeX vai interpretar que todas essas linhas fazem parte do mesmo parágrafo.

Embora isso seja estranho, algumas pessoas preferem trabalhar assim para terem controle sobre cada frase.

E o que acontece se deixarmos várias linhas em brancos? Vamos testar?

```
\end{document}
```

## **lutex05.tex**

```
\documentclass{article}
```

```
\usepackage{xltxtra}
```

```
\begin{document}
```

Uma frase com vários espaços diferentes entre as palavras.

Uma frase com apenas um espaço entre as palavras

```
\end{document}
```

## **lutex06.tex**

```
\documentclass{article}
```

```
\usepackage{xltxtra}
```

```
\begin{document}
```

No LaTeX, é “muito” importante lembrar de fechar as aspas.

```
\end{document}
```

## **lutex07a.tex**

```
\documentclass{article}
\usepackage{xltextra}
\begin{document}
```

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

```
\end{document}
```

## **lutex07b.tex**

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{polyglossia}
\setmainlanguage[brazil]
\begin{document}
```

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

```
\end{document}
```

## **lutex07c.tex**

```
\documentclass[abntex2]{abntex2}
\usepackage{xltextra}
\begin{document}
```

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

```
\end{document}
```

## **lutex07d.tex**

```
\documentclass{article}  
\usepackage{xltextra}  
\hyphenation{al-te-ra-das}  
\begin{document}
```

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

```
\end{document}
```

## **lutex07e.tex**

```
\documentclass{article}  
\usepackage{xltextra}  
\hyphenation{alteradas automaticamente}  
\begin{document}
```

Sabemos que o LaTeX se encarrega de organizar os parágrafos automaticamente. Mas e se houver muitas palavras bastante longas? Como elas serão alteradas? Por exemplo, vejamos a seguinte lista: otorrinolaringologista, proparoxítonas, estereotipadamente, espécime, crisântemo, hieróglifo, inconstitucionalíssimo.

```
\end{document}
```

## **lutex08.tex**

```
\documentclass[14pt,oneside]{article}
\usepackage{xltxtra}
\begin{document}
Seu texto.
\end{document}
```

## **lutex09a.tex**

```
\documentclass{article} % Troque por “book” e veja as diferenças.
\usepackage{xltxtra}
\author{Nome do autor}
\title{Título do documento}
\date{\today}
\begin{document}
\maketitle % O comando que diz para o LaTeX gerar um título com as informações do
% preâmbulo.
Seu texto.
\end{document}
```

## **lutex09b.tex**

```
\documentclass{book} % Troque por “article” e veja as diferenças.
\usepackage{xltxtra}
\author{Nome do autor}
\title{Título do documento}
\date{\today}
\begin{document}
\maketitle % O comando que diz para o LaTeX gerar um título com as informações do
% preâmbulo.
```

```
Seu texto.  
\end{document}
```

## **lutex10a.tex**

```
\documentclass[article]{  
\\usepackage{xltxtra}  
\\title{Exemplo “LuTeX10a”: criando uma estrutura para seu texto}  
\\author{Lucas Mafaldo}  
\\date{\\today}  
\\begin{document}  
\\maketitle  
\\section{Título da primeira seção} % Cria uma nova seção.  
\\subsection{Título da primeira subseção} % Cria uma seção subordinada à seção criada anteriormente.  
Conteúdo da \\emph{primeira} subseção % O leitor ainda lembra do comando “\\emph{}”?  
\\subsection{Título da \\emph{segunda} subseção} % Podemos utilizar códigos de formatação nos títulos.  
Conteúdo da segunda subseção  
\\section{Nova seção}  
Conteúdo da nova seção.  
\\end{document}
```

## **lutex11a.tex**

```
\documentclass[oneside]{book} % Modificamos essa linha, trocando “article” por “book” e adicionamos uma nova opção.  
\\usepackage{xltxtra}  
\\title{Exemplo “LuTeX11a”: criando uma estrutura para seu livro} % Novo título.  
\\author{Lucas Mafaldo}
```

```
\date{\today}
\begin{document}
\maketitle
\part{Início do livro}
\chapter{Conteúdo do arquivo LuTeX10a.tex}
\section{Título da primeira seção} % Cria uma nova seção.
\subsection{Título da primeira subseção} % Cria uma seção subordinada à seção criada anteriormente.
    Conteúdo da \emph{primeira} subseção % O leitor ainda lembra do comando “\emph{}”?
\subsection{Título da \emph{segunda} subseção} % Podemos utilizar códigos de formatação nos títulos.
    Conteúdo da segunda subseção
\section{Nova seção}
    Conteúdo da nova seção.
\chapter{Capítulo com conteúdo novo}
    Conteúdo do segundo capítulo.
\end{document}
```

## lutex10b.tex

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{polyglossia} % Novo pacote de línguas.
\setmainlanguage[brazil] % Especifica que a língua utilizada será o português brasileiro.
\title{Exemplo “LuTeX10b”: gerando um sumário a partir da estrutura do documento} % Novo título
\author{Lucas Mafaldo}
\date{\today}
\begin{document}
\maketitle
\tableofcontents % Comando que será utilizado para gerar o sumário.
```

```
\section{Título da primeira seção}

\subsection{Título da primeira subseção}

Conteúdo da \emph{primeira} subseção.

\subsection{Título da \emph{segunda} subseção}

Conteúdo da segunda subseção.

\section{Nova seção}

Conteúdo da nova seção.

\end{document}
```

## **lutex11b.tex**

```
\documentclass[oneside]{book} % Modificamos essa linha, trocando “article” por “book” e
% adicionamos uma nova opção.

\usepackage{xltextra}
\usepackage{polyglossia} % Novo pacote de línguas.
\setmainlanguage[brazil]

\title{Exemplo “LuTeX11b”: gerando um sumário para seu livro} % Novo título.

\author{Lucas Mafaldo}
\date{\today}

\begin{document}
\maketitle

\tableofcontents % Comando que “diz” para o LaTeX gerar automaticamente um sumário.

\part{Início do livro}

\chapter{Conteúdo do arquivo LuTeX10a.tex}

\section{Título da primeira seção}

\subsection{Título da primeira subseção}

Conteúdo da \emph{primeira} subseção.

\subsection{Título da \emph{segunda} subseção}

Conteúdo da segunda subseção.

\section{Nova seção}

Conteúdo da nova seção.
```

```
\chapter{Capítulo com conteúdo novo}
Conteúdo do segundo capítulo.
\end{document}
```

## **lutex12.tex**

```
\documentclass{article} % Alterne a classe entre “abntex2” e “article” e compare os resultados.
\usepackage{xltxtra}
\begin{document}
Texto normal. Texto normal. Texto normal. Texto normal. Texto normal. Texto normal. Texto
normal. Texto normal.
\begin{quote}
Texto citado. Texto citado. Texto citado. Texto citado. Texto citado. Texto citado. Texto citado.
Texto citado. Texto citado. Texto citado. Texto citado. Texto citado.
\end{quote}
\end{document}
```

## **lutex13.tex**

```
\documentclass{article}
\usepackage{xltxtra}
\begin{document}
\begin{enumerate}
\item Primeiro ítem;
\item Segundo ítem.
\end{enumerate}
\begin{itemize}
\item Um ítem não-numerado.
\item Outro ítem não-numerado
\end{itemize}
\end{document}
```

```
\begin{description}
\item[enumerate] é o código para criar listas numeradas.
\item[itemize] é o código para criar listas não-numeradas.
\item[description] é o código para criar listas descritivas.
\end{description}
\end{document}
```

## **lutex14.tex**

```
\documentclass{article}
\usepackage{xltextra}
\begin{document}
\begin{flushleft} Todo o texto desse ambiente ficará alinhado à esquerda. \end{flushleft}
\begin{flushright} Todo o texto desse ambiente ficará alinhado à direita. \end{flushright}
\begin{center} Todo o texto desse ambiente ficará centralizado. \end{center}
\end{document}
```

## **lutex15a.tex**

```
\documentclass{article}
\usepackage{xltextra}
\begin{document}
Esse é o texto do parágrafo. No final dessa frase, iremos inserir uma nota de rodapé, com a marcação diretamente anterior ao ponto final\footnote{Esse é o texto da nota de rodapé.}. Apesar da nota de rodapé já ter sido inserida, o parágrafo continua.
\end{document}
```

## **lutex15b.tex**

```
\documentclass{article}
\usepackage{xltxtra}
\newcommand\fn[1]{\footnote{\#1}}
\begin{document}

Esse é o texto do parágrafo. No final dessa frase, iremos inserir uma nota de rodapé, com a marcação diretamente anterior ao ponto final\fn{Esse é o texto da nota de rodapé.}. Apesar da nota de rodapé já ter inserida, o parágrafo continua.

\end{document}
```

## bibliografia.bib

```
@book{gonzalez-dialectic,
Address = {Evanston, Illinois},
Author = {Francisco J. Gonzalez},
Publisher = {Northwestern University Press},
Title = {Dialectic and Dialogue: Plato's Practice of Philosophical Inquiry},
Year = {1998}}
@article{plato-rhetoric,
Author = {Edward Schiappa},
Journal = {The American Journal of Philology},
Number = {4},
Pages = {457-470},
Title = {Did Plato Coin Rhetorike},
Volume = {111},
Year = {1990}}
```

## lutex16a.tex

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{natbib} % Comando que diz para o LaTeX carregar o pacote “natbib”.
\begin{document}
Seu texto.

\bibliography{bibliografia} % Dizemos ao LaTeX qual base de dados bibliográficos utilizar.
\bibliographystyle{plainnat} % Dizemos ao LaTeX para formatar a bibliografia segundo o estilo “plainnat”.
\end{document}
```

## **lutex16b.tex**

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{natbib}
\begin{document}
É provável que Platão tenha inventado o termo “retórica”. \citet{plato-rhetoric}
apresenta uma interpretação interessante sobre a
\emph{Carta Sétima} de Platão
\bibliography{bibliografia}
\bibliographystyle{plainnat}
\end{document}
```

## **lutex16c.tex**

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{polyglossia}
\setmainlanguage[brazil]
```

```
\usepackage{natbib}
\begin{document}
É provável que Platão tenha inventado o termo “retórica”. \citep{plato-rhetoric}
\citet[p. 245-274]{gonzalez-dialectic} apresenta uma interpretação interessante sobre a
\emph{Carta Sétima} de Platão
\bibliography{bibliografia}
\bibliographystyle{plainnat}
\end{document}
```

## **lutex16d.tex**

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{polyglossia}
\setmainlanguage[brazil]
\usepackage{natbib}
\begin{document}
Um texto sem referências.
\nocite{plato-rhetoric}
\bibliography{bibliografia}
\bibliographystyle{plainnat}
\end{document}
```

## **lutex16e.tex**

```
\documentclass{article}
\usepackage{xltextra}
\usepackage{polyglossia}
\setmainlanguage[brazil]
```

```
\usepackage{natbib}  
\begin{document}  
Um texto sem referências.  
\nocite{*}  
\bibliography{bibliografia}  
\bibliographystyle{plainnat}  
\end{document}
```

## **lutex17a.tex**

```
\documentclass{article}  
\usepackage{xltxtra}  
\usepackage[alf]{abntex2cite}  
\begin{document}  
É provável que Platão tenha inventado o termo “retórica”. \cite{plato-rhetoric}  
\citetext[p.~245–274]{gonzalez-dialectic} apresenta uma interpretação interessante sobre a  
\emph{Carta Sétima} de Platão  
\bibliography{bibliografia}  
\end{document}
```

## **lutex17b.tex**

```
\documentclass[article]{abntex2} % Observem que “article” é uma opção da classe “abntex2”.  
\usepackage{xltxtra}  
\usepackage[alf]{abntex2cite}  
\begin{document}  
É provável que Platão tenha inventado o termo “retórica”. \cite{plato-rhetoric}  
\citetext[p.~245–274]{gonzalez-dialectic} apresenta uma interpretação interessante sobre a  
\emph{Carta Sétima} de Platão
```

```
\bibliography{bibliografia}  
\end{document}
```

## **lutax18a.tex**

```
\documentclass[10pt]{beamer}  
\usepackage{xltextra}  
\usetheme{Frankfurt}  
\usecolortheme{dove}  
\usepackage{polyglossia}  
\setmainlanguage{brazil}  
\title[Um subtítulo]{O título principal}  
\author{Lucas Mafaldo}  
\institute{Academia LuTeX}  
\date{\today}  
\begin{document}  
\section{Introdução}  
\begin{frame}  
\titlepage  
\end{frame}  
\begin{frame}  
\tableofcontents  
\end{frame}  
\section{Discussão dos dados}  
\subsection{Dados encontrados}  
\begin{frame}  
Encontramos os seguintes dados:  
\begin{itemize}  
\item Dados A. \pause  
\item Dados B.  
\end{itemize}
```

```
\end{frame}
\subsection{Análise dos dados}
\begin{frame}
Texto \emph{desta} subseção.
\end{frame}
\section{Considerações finais}
\begin{frame}
Conteúdo desta seção\footnote{Esta é a última seção}.
\end{frame}
\end{document}
```

## **lutex18b.tex**

```
\documentclass[10pt]{beamer}
\usepackage{xltxtra}
\usetheme{Amsterdam}
\usepackage{polyglossia}
\setmainlanguage[brazil]
\title[Um subtítulo]{O título principal}
\author{Lucas Mafaldo}
\institute{Academia LuTeX}
\date{\today}
\begin{document}
\section{Introdução}
\begin{frame}
\titlepage
\end{frame}
\begin{frame}
\tableofcontents
\end{frame}
\end{document}
```

```
\section{Discussão dos dados}

\subsection{Dados encontrados}

\begin{frame}
Encontramos os seguintes dados:
\begin{itemize}
\item Dados A. \pause
\item Dados B.
\end{itemize}
\end{frame}

\subsection{Análise dos dados}

\begin{frame}
Texto \emph{desta} subseção.
\end{frame}

\section{Considerações finais}

\begin{frame}
Conteúdo desta seção\footnote{Esta é a última seção}.
\end{frame}

\end{document}
```