Texinfo

O Formato da Documentação GNU
para Texinfo versão 6.1, 06 de fevereiro de 2016

Robert James Chassell
Richard Matthew Stallman

Este manual é para GNU Texinfo (versão 6.1, 06 de fevereiro de 2016), um sistema de documentação que pode produzir ambos informação online e um manual impresso a partir de uma fonte única usando marcação semântica.

This manual is for GNU Texinfo (version 6.1, 06 de fevereiro de 2016), a documentation system that can produce both online information and a printed manual from a single source using semantic markup.

Publicado pela Free Software Foundation
51 Franklin St, Fifth Floor
Boston, MA 02110-1301
USA
ISBN 1-882114-67-1

Arte da capa por Etienne Suvasa.

Breve Sumário

Sumário

Documentação é como o sexo: quando ela é boa, ela é muito, muito boa; e quando ela é ruim, é melhor que nada. —Dick Brandon

Condições de Cópia do Texinfo

GNU Texinfo é *software livre*; isso significa que qualquer pessoa é livre para usá-lo e livre para redistribuí-lo sob certas condições. Texinfo não está em domínio público; ele está sob direitos autorais e existem restrições acerca da distribuição dele; porém essas restrições estão projetadas para permitir qualquer coisa que um(a) bom(a) cidadão(ã) cooperador(a) desejasse fazer. O que não for permitido é para tentar impedir que outros(as) adicionalmente compartilhem qualquer versão de Texinfo que eles(as) possam obter de você.

Especificamente, nós queremos ter certeza que você tem o direito de doar cópias dos programas que se relacionem a Texinfo; que você receba código fonte ou, do contrário, possa obtê-lo se assim o desejar; que você pode mudar esses programas ou usar pedaços deles em novos programas livres; e que você sabe que pode fazer essas coisas.

Para ter certeza de que qualquer pessoa tem tais direitos, nós temos que proibir que você prive alguém mais desses direitos. Por exemplo, se você distribuir cópias dos programas relacionados a Texinfo, você precisa dar para os(as) receptores(as) todos os direitos que você tem. Você precisa assegurar-se de que eles(as), também, recebam ou possam conseguir o código fonte. E você precisa informá-los(as) dos direitos deles(as).

Além disso, para nossa própria proteção, nós devemos estar certos(as) de que qualquer pessoa sabe que não existe garantia para os programas que se relacionem a Texinfo. Se esses programas forem modificados por qualquer outra pessoa e passados adiante, nós queremos que os(as) receptores(as) deles saibam que o que eles(as) tem não é o que nós distribuímos, de forma que quaisquer problemas introduzidos por outros(as) não refletirão na nossa reputação.

As condições precisas das licenças para os programas atualmente sendo distribuídos que se relacionem com Texinfo são encontradas nas Licenças Gerais Públicas que os acompanham. Este manual está coberto pela Licença GNU de Documentação Livre (veja Apêndice H [Licença GNU de Documentação Livre], Página 265).

1 Visão Geral do Texinfo

Texinfo é um sistema de documentação que usa um arquivo fonte único para produzir informações online e saída impressa. Isso significa que, em vez de escrever vários documentos, um para cada formato de saída, você precisa escrever somente um documento.

Usando Texinfo, você consegue criar um documento impresso (via sistema tipográfico TEX) em formato PDF ou PostScript, incluindo capítulos, seções, referências cruzadas e índices. A partir do mesmo arquivo fonte do Texinfo, você consegue criar um arquivo de saída HTML adequado para uso com um navegador web; você consegue criar um arquivo do Info com recursos especiais para facilitar a navegação da documentação; e também criar um arquivo do Docbook ou uma transliteração para o formato XML.

Um arquivo fonte do Texinfo é um arquivo de texto simples contendo texto intercalado com *comandos* @ (palavras precedidas por um '`@`') que informam aos processadores Texinfo o que fazer. Os comandos de marcação do Texinfo são quase inteiramente *semânticos*; isto é, eles especificam o significado pretendido do texto no documento, em vez de instruções de formatação física. Você consegue editar um arquivo do Texinfo com qualquer editor de texto, porém é especialmente conveniente se usar o GNU Emacs, dado que esse editor tem um modo especial, chamado modo Texinfo, que fornece vários recursos relacionados ao Texinfo (Veja Apêndice D [Modo Texinfo], Página 237).

Texinfo foi concebido especificamente para o propósito de escrever documentação e manuais de software. Se você quer escrever um bom manual para teu programa, Texinfo tem muitos recursos que nós esperamos que tornarão tua tarefa mais fácil. Entretanto, ele quase não fornece comandos para controlar a formatação final. Texinfo não é destinado a ser um programa de formatação de propósito geral, de modo que se você precisa planejar um jornal, conceber um anúncio luxuoso de revista, ou seguir as exatas exigências de formatação de uma editora, Texinfo pode não ser a ferramenta mais simples.

Escreva-se "Texinfo" com um "T" maiúsculo e as outras letras em minúsculas. A primeira silaba de "Texinfo" é pronunciada como "speck", e não "hex". Essa pronúncia estranha é derivada da pronúncia de TEX. Pronuncie-se TEX como se o '`X`' fosse o último som no nome 'Bach'. Na palavra TEX, o '`X`' é, em vez da letra do Inglês "ex", na verdade a letra Grega "chi".

Texinfo é o formato oficial de documentação do projeto GNU. Mais informações, incluindo manuais para pacotes GNU, estão disponíveis na página web da documentação GNU (`http://www.gnu.org/doc/`).

1.1 Informando Defeitos

Nós apreciamos informes de defeitos e de sugestões para qualquer aspecto do sistema Texinfo: programas, documentação, instalação, etc. Por favor, envie-os para `bug-texinfo@gnu.org`. Você consegue obter a versão mais recente do Texinfo por meio da página inicial dele, `http://www.gnu.org/software/texinfo`.

Para informes de defeitos, por favor, inclua informações suficientes para os(as) mantenedores(as) reproduzirem o problema. Falando genericamente, isso significa:

- O número da versão do Texinfo e o(s) programa(s) ou manual(is) envolvido(s).
- O conteúdo de quaisquer arquivos de entrada necessários para reproduzir o defeito.
- Precisamente como você executou quaisquer programa(s) envolvido(s).
- Uma descrição do problema e amostras de quaisquer saídas errôneas.
- Nomes e versões do hardware e do sistema operacional.
- Qualquer outra coisa que você pense que pudesse ajudar.

Quando em dúvida se alguma coisa é necessária ou não, inclua-a. É melhor incluir muito mais que deixar de fora algo importante.

É crítico enviar um arquivo de entrada atual que reproduza o problema. O que não é crítico é "simplificar" o exemplo para a menor entrada possível—a entrada atual com a qual você descobriu o defeito bastará. (Certamente, se você fizer experimentos, quanto menor o arquivo de entrada, melhor).

Correções ("Patches") são mais que bem-vindas; se possível, por favor faça-as com '`diff -c`' (veja-se *Comparing and Merging Files*) e inclua as entradas `ChangeLog` (veja-se Seção "Change Log" em *O Manual do GNU Emacs*), e siga o estilo de codificação existente.

1.2 Formatos de Saída

Aqui está uma visão geral breve dos formatos de saída atualmente suportados por Texinfo.

Info (Gerado via `makeinfo`). O formato Info é em sua maior parte transliteração de texto plano do fonte Texinfo. Ele adiciona uns poucos caracteres de controle para prover informação de navegação para referências cruzadas, índices, e assim por diante. O subsistema Emacs Info (veja-se *Info*), e o programa autônomo `info` (veja *GNU Info*), entre outros, podem ler esses arquivos. Veja-se Seção 1.3 [Arquivos do Info], Página 5, e Capítulo 21 [Criando e Instalando Arquivos Info], Página 185.

Texto simples
(Gerado via `makeinfo --plaintext`). Essa é quase a mesma que a saída Info com os caracteres de controle de navegação omitidos.

HTML (Gerado via `makeinfo --html`). HTML, significando Hyper Text Markup Language (Linguagem de Marcação de Hiper Texto), tem se tornado a mais comumente usada linguagem para a escrita de documentos na World Wide Web (Teia de Alcance Global). Os navegadores web, tais como Mozilla, Lynx, e Emacs-W3, podem renderizar essa linguagem online. Existem muitas versões de HTML, também padrões diferentes e variações específicas de navegador. `makeinfo` tenta usar um subconjunto da linguagem que possa ser interpretado por qualquer navegador comum, intencionalmente não se usando de muitas marcações mais novas ou menos amplamente suportadas. Apesar que a saída nativa é assim até certo ponto plana, ela pode ser personalizada em vários níveis, se desejado. Para detalhes da linguagem HTML e mais informação relacionada, veja-se `http://www.w3.org/MarkUp/`. Veja-se Capítulo 22 [Gerando HTML], Página 194.

DVI (Gerado via `texi2dvi`). O formato binário "DeVIce Independent" é liberado pelo programa tipográfico TEX (`http://tug.org`). Essa é então lida por um 'controlador' DVI, o qual conhece os comandos específicos de dispositivo atuais que podem ser visualizados ou impressos, notadamente Dvips para tradução para PostScript (veja-se *Dvips*) e Xdvi para visualização em uma tela X (`http://sourceforge.net/projects/xdvi/`). Veja-se Capítulo 19 [Impresso], Página 150. (Esteja avisado de que a linguagem Texinfo é muito diferente de, e muito mais estrita que, as linguagens usuais de TEX: plain TEX, LATEX, ConTEXt, etc.).

PostScript (Gerada via `texi2dvi --ps`). PostScript é uma linguagem de descrição de página que se tornou amplamente usada por volta de 1985 e ainda é usada hoje em dia. `http://en.wikipedia.org/wiki/PostScript` dá uma descrição básica e mais preferências. Por padrão, Texinfo usa o programa `dvips` para converter a saída DVI de TEX para PostScript. Veja-se *Dvips*.

PDF (Gerada via `texi2dvi --pdf` ou `texi2pdf`). Esse formato foi desenvolvido por Adobe Systems para troca de documento portável, baseado na linguagem PostScript

prévia deles. O formato pode representar a exata aparência de um documento, incluindo fontes e gráficos, e suportar escalonamento arbitrário. Ele é entendido como sendo independente de plataforma e facilmente visualizável, entre outros objetivos de design; `http://en.wikipedia.org/wiki/Portable_Document_Format` e `http://tug.org/TUGboat/tb22-3/tb72beebe-pdf.pdf` tem algum conhecimento. Por padrão, Texinfo usa o programa `pdftex`, uma extensão de TEX, para liberar PDF; veja-se `http://tug.org/applications/pdftex`. Veja-se Seção 19.15 [Saída PDF], Página 161.

Docbook (Gerado via `makeinfo --docbook`). Esse é um formato baseado em XML desenvolvido alguns anos atrás, primariamente para documentação técnica. Ele por conseguinte assume alguma semelhança, em linhas gerais, com Texinfo. Veja-se `http://www.docbook.org`. Vários conversores de Docbook *para* Texinfo também foram desenvolvidos; veja-se as páginas web Texinfo.

XML (Gerado via `makeinfo --xml`). XML é uma especificação de sintaxe genérica usável para qualquer tipo de conteúdo (uma referência está em `http://www.w3.org/XML`). A saída XML de `makeinfo`, diferente de todos os outros formatos de saída, é uma transliteração do fonte Texinfo em vez de saída processada. Isto é, a saída traduz os comandos de marcação Texinfo em sintaxe XML, para processamento mais amplo por ferramentas XML. Os detalhes da saída estão definidos em um DTD XML como de costume, o qual está contido em um arquivo `texinfo.dtd` incluso na distribuição do fonte Texinfo e disponível via páginas web Texinfo. O XML contém informação suficiente para recriar o conteúdo original, exceto para construções sintáticas, tais como macros Texinfo e condicionais. A distribuição do fonte Texinfo inclui um script utilitário `txixml2texi` para fazer essa transformação de volta.

1.3 Arquivos do Info

Conforme mencionado acima, o formato Info é em sua maioria uma transliteração de texto plano do fonte Texinfo, com a adição de uns poucos caracteres de controle para separar nós e prover informação de navegação, de forma que os programas de leitura Info possam operar sobre ele.

Os arquivos Info são quase sempre criados pelo processamento de um documento fonte Texinfo. O comando `makeinfo`, também conhecido como `texi2any`, é o comando principal que converte um arquivo Texinfo em um arquivo Info; veja-se Capítulo 20 [Tradutor Genérico `texi2any`], Página 162.

Geralmente, você adentra um arquivo Info via um nó que, por convenção, é chamado 'Top'. Esse nó normalmente contém somente um sumário curto do propósito do arquivo, e um menu amplo por meio do qual o restante do arquivo é alcançado. A partir desse nó, você ou pode atravessar o arquivo sistematicamente, indo de nó a nó; ou você pode ir até um nó específico listado no menu principal, ou você pode pesquisar no menu de índice e então ir diretamente ao nó que tenha a informação que você deseja. Alternativamente, com programa autônomo Info, você pode especificar itens de menu na linha de comando (veja-se *Info*).

Se você desejar ler ao longo de um arquivo Info em sequência, como se ele fosse um manual impresso, você pode teclar `ESPAÇO` repetidamente, ou você percorre o arquivo inteiro com o comando Info avançado *g* `*`. (Veja-se Seção "Advanced Info commands" em *Info*).

O arquivo `dir` no diretório `info` serve como o ponto de chegada para o sistema Info inteiro. A partir dele, você pode alcançar os nós 'Top' de cada um dos documentos em um sistema Info completo.

Se você desejar se referir a um arquivo Info via URI, você pode usar a sintaxe (não oficial) exemplificada pelo seguinte. Isto funciona com Emacs/W3, por exemplo:

```
info:emacs#Dissociated%20Press
```

```
info:///usr/info/emacs#Dissociated%20Press
info://localhost/usr/info/emacs#Dissociated%20Press
```

O próprio programa `info` não segue URIs de qualquer tipo.

1.4 Livros Impressos

Um arquivo Texinfo pode ser formatado e composto como um livro impresso ou um manual. Para fazer isso, você precisa de TEX, um sofisticado programa de tipografia escrito por Donald Knuth da Universidade Stanford.

Um livro baseado em Texinfo é similar a qualquer outro composto, trabalho impresso: ele pode ter uma página de título, página de direitos autorais, sumário, e prefácio, bem como capítulos, seções e subseções numeradas ou não numeradas, cabeçalhos de páginas, referências cruzadas, notas de rodapé, e índices.

TEX é um programa tipográfico de propósito geral. Texinfo provê um arquivo `texinfo.tex` que contém informação (definições ou *macros*) que TEX usa quando compõe arquivo Texinfo. (`texinfo.tex` informa a TEX como converter os comandos @ de Texinfo para comandos TEX, os quais TEX pode então processar para criar o documento tipografado). `texinfo.tex` contém as especificações para imprimir um documento. Você pode obter a versão mais recente de `texinfo.tex` a partir da página de Texinfo `http://www.gnu.org/software/texinfo/`.

Nos Estados Unidos, os documentos são em sua maioria frequentemente impressos em páginas de 8.5 por 11 polegadas (216 mm por 280 mm); esse é o tamanho padrão. Porém, você também pode imprimir em páginas de 7 por 9.25 polegadas (178 mm por 235 mm, o tamanho `@smallbook`; ou em papel de tamanho A4 ou A5 (`@afourpaper`, `@afivepaper`). Veja-se Seção 19.11 [`@smallbook`], Página 159, e Seção 19.12 [Papel A4], Página 159.

TEX é livremente distribuível. Ele é escrito em um super conjunto de Pascal para programação de instrução chamado WEB e pode ser compilado ou em Pascal ou (pelo uso de um programa de conversão que vem com a distribuição de TEX) em C.

TEX é muito poderoso e tem um grande número de características. Porque um arquivo Texinfo deve necessariamente estar apto a apresentar informação tanto em um terminal somente caractere em formato Info quanto em um livro tipografado, os comandos de formatação que Texinfo suporta são necessariamente limitados. Veja-se Seção 19.16 [Obtendo TEX], Página 161, para informação sobre como adquirir TEX. Ele não é parte da distribuição Texinfo.

1.5 Adicionando Formatos de Saída

Os formatos de saída nas seções anteriores lidam com uma variedade ampla de usos, porém, certamente, sempre existe espaço para mais.

Se você for um programador e gostaria de contribuir para com o projeto GNU implementando formatos de saída adicionais para Texinfo, isso seria excelente. A maneira de fazer isso que seria mais útil é escrever uma infraestrutura nova para `texi2any`, nossa implementação de referência de um analisador Texinfo; ela cria uma representação de árvore da entrada de Texinfo que você pode usar para a conversão. A documentação no arquivo fonte `tp/Texinfo/Convert/Converter.pm` é um bom lugar para começar. Veja-se Capítulo 20 [Tradutor Genérico `texi2any`], Página 162.

Outra abordagem viável é usar a saída XML de Texinfo oriunda de `texi2any` como sua entrada. Esse XML é uma representação essencialmente completa da entrada, porém sem a sintaxe de Texinfo e as peculiaridades de opção, conforme descrito acima.

Se você ainda não conseguir resistir à tentação de escrever um programa novo que leia o fonte Texinfo diretamente, permita-nos te dar mais algumas advertências: por favor não subestime a quantidade de trabalho exigida. Texinfo não é de forma alguma uma linguagem simples de se analisar corretamente, e permanece sob desenvolvimento, de maneira que você estaria se

comprometendo com uma tarefa em andamento. Você está aconselhado a verificar se os testes da linguagem que vem com `texi2any` dão resultados corretos com o seu programa novo.

De tempos em tempos, propostas são feitas para gerar páginas de manual Unix tradicionais a partir do fonte Texinfo. Entretanto, pelo motivo de que as páginas de manual tem um formato convencional estrito, criar uma página de manual boa exige um fonte completamente diferente daquele necessário para as aplicações Texinfo típicas de escrita de um tutorial de usuário bom e/ou um bom manual de referência. Isso torna a geração de páginas de manual incompatível com o objetivo do desenho de Texinfo de não ter de documentar a mesma informação em maneiras diferentes para diferentes formatos de saída. Você poderia também escrever a página de manual diretamente.

Como um meio alternativo para suportar páginas de manual, você pode achar que o programa `help2man` seja útil. Ele gera uma página de manual tradicional a partir da saída '`--help`' de um programa. De fato, as páginas de manual para os programas na distribuição Texinfo são geradas com isso. Ele é software GNU escrito por Brendan O'Dea, disponível a partir de `http://www.gnu.org/software/help2man`.

1.6 Histórico

Richard M. Stallman inventou o formato Texinfo, escreveu os processadores iniciais, e criou a Edição 1.0 deste manual. Robert J. Chassell revisou e estendeu imensamente o manual, iniciando com a Edição 1.1. Brian Fox foi responsável pela distribuição autônoma de Texinfo até a versão 3.8, e originalmente escreveu os programas autônomos `makeinfo` e `info`. Karl Berry continuou a manutenção desde Texinfo 3.8 (edição do manual 2.22).

Nossos agradecimentos vão para todos que ajudaram a aperfeiçoar este trabalho, particularmente os incansáveis Eli Zaretskii e Andreas Schwab, que forneceram correções incontáveis. François Pinard e David D. Zuhn, incansavelmente gravaram e relataram erros e obscuridades. Zack Weinberg fez o impossível implementando a sintaxe de macro em `texinfo.tex`. Obrigado a Melissa Weisshaus por suas frequentes revisões de edições quase similares. Dúzias de outros contribuíram com correções e sugestões, eles estão agradecidamente reconhecidos no arquivo `ChangeLog`. Nossos erros são nossos próprios.

Começo

Nos anos 1970 no CMU, Brian Reid desenvolveu um programa e formato chamado Scribe para marcar documentos para impressão. Ele usou o caractere `@` para introduzir comandos, conforme Texinfo faz. Muito mais consequencialmente, ele se empenhou em descrever o conteúdo dos documentos em vez da formatação, uma ideia inteiramente adotada por Texinfo.

Ao mesmo tempo, pessoas no MIT desenvolveram outro, não muito diferente formato chamado Bolio. Esse então foi convertido usando TEX como sua linguagem tipográfica: BoTEX. A versão mais antiga de BoTEX parece ter sido a 0.02 em 31 de outubro de 1984.

BoTEX somente poderia ser usado como uma linguagem de marcação para documentos serem impressos, não para documentos online. Richard Stallman (RMS) trabalhou em ambos Bolio e BoTEX. Ele também desenvolveu um formato chique de ajuda on-line chamado Info, e então combinou BoTEX e Info para criar Texinfo, uma linguagem de marcação para texto que é entendida para ler ambos online e como cópia impressa.

Seguindo em frente, o tradutor original para criar Info foi escrito (primariamente por RMS e Bob Chassell) em Emacs Lisp, a saber: o `texinfo-format-buffer` e outras funções. No começo dos anos 1990, Brian Fox reimplementou o programa de conversão em C, agora chamado `makeinfo`.

Reimplementando in Perl

Em 2012, o `makeinfo` C foi ele próprio substituído por uma implementação Perl chamada genericamente `texi2any`. Essa versão suporta o mesmo nível de personalização de saída que `texi2html`, um programa independente originalmente escrito por Lionel Cons, mais tarde com trabalho substancial de muitos outros. As muitas características adicionais necessárias para tornar `texi2html` uma substituição para `makeinfo` foram implementadas por Patrice Dumas. A primeira versão jamais lançada de `texi2any` foi baseada no código `texi2html`. Essa implementação, entretanto, foi abandonada em favor do atual programa, que analisa a entrada Texinfo em uma árvore para processamento. Ele ainda suporta quase todas as características de `texi2html`.

O novo programa Perl é muito mais lento que o antigo programa C. Nós esperamos que a diferença de velocidade será fechada no futuro, porém elas podem jamais ser inteiramente comparáveis. Então porque nós mudamos? Em poucas palavras, nós pretendemos e esperamos que o presente programa seja muito mais fácil que a prévia implementação C de `makeinfo` para estender para diferentes estilos de saída, formatos de infra estrutura de saída, e todas as outras personalizações. Em mais detalhes:

- Personalização de HTML. Muitos GNU e outros pacotes de software livre tem sido felizes no uso das características de personalização de HTML em `texi2html` por anos. Assim, na verdade, duas implementações independentes da linguagem Texinfo foram desenvolvidas, e mantê-las em sincronia não foi simples. Adicionar a possibilidade de personalização de HTML em `texi2html` a um programa C tem sido um esforço enorme.
- Unicode, e suporte multilínguas genericamente, especialmente das linguagens da Ásia Oriental. Apesar que certamente é perfeitamente plausível escrever tal suporte em C, no caso particular de `makeinfo`, isso teria sido equivalente a reescrever o programa inteiro. Em Perl, muito disso vem essencialmente de graça.
- Infra estrutura adicional. O código `makeinfo` se tornou tão complexo ao ponto de que, adicionar uma infra estrutura nova, era bastante complexo, exigindo interações complexas com as infra estruturas existentes. Em contraste, nossa implementação Perl provê uma representação limpa baseada em árvore para todas as infra estruturas funcionarem. Pessoas tem solicitado numerosas infra estruturas diferentes (LaTeX, o mais recente (X)HTML, ...), e elas agora serão muito mais viáveis de implementar. O que conduz ao último item:
- Tornando as contribuições mais fáceis. Em geral, devido à estrutura mais limpa, o programa Perl deveria ser consideravelmente mais fácil que o C para qualquer pessoa ler e contribuir com, com os óbvios benefícios resultantes.

Veja-se Seção 20.1 [Implementação de Referência], Página 162, para mais sobre a razão de ser e função de `texi2any`.

2 Escrevendo um Arquivo do.Texinfo

Este capítulo descreve a sintaxe Texinfo e o que é exigido em um arquivo Texinfo, e dá um arquivo curto de amostra.

2.1 Convenções Sintáticas Gerais

Esta seção descreve as convenções gerais usadas em todos os documentos Texinfo.

- Todos os caracteres ASCII imprimíveis, exceto '`@`', '`{`' e '`}`', podem aparecer em um arquivo Texinfo e representam eles mesmos. '`@`' é o caractere de encapsulamento que introduz comandos, enquanto '`{`' e '`}`' são usados para envolver argumentos a certos comandos. Para colocar um desses caracteres especiais no documento, coloque um caractere '`@`' em frente a ele, como isto: '`@@`', '`@{`', e '`@}`'.
- Em um arquivo Texinfo, os comandos que você escreve para descrever o conteúdo do manual são precedidos por um caractere '`@`'; eles são chamados *comandos @*. (O '`@`' em Texinfo tem o mesmo significado que '\' tem em TEX plano).

  Dependendo do que fazem ou quais argumentos[1] receberem, você precisa escrever comandos @ em suas próprias linhas, ou como parte de sentenças. Como uma regra geral, um comando exige chaves se esse comando se misturar entre outro texto; porém, ele não necessita das chaves se estiver em sua própria linha. Para mais detalhes da sintaxe de comando Texinfo, veja-se Seção A.1 [Sintaxe de Comando], Página 204.
- Espaço em branco seguindo um nome de comando @ é opcional e (normalmente) ignorado se presente. As exceções são contextos quando o espaço em branco é significativo, por exemplo, um ambiente de `@example`.
- Texinfo suporta as marcações normais de encapsulamento usados em Inglês e em outras linguagens; veja-se Seção 12.5 [Inserindo Aspas], Página 101.
- Use três hifens em uma linha, '`---`', para produzir um traço longo—como esse (chamado um *travessão*), usado para pontuação em sentenças. Use dois hifens, '`--`', para produzir um traço médio (chamado um *traço de ligação*), usado primariamente para intervalos numéricos, como em "Junho 25–26". Use um hífen simples, '`-`', para produzir um hífen padrão usado em palavras compostas. Para exibir na tela, Info reduz três hifens para dois e dois hifens para um (não transitivamente!). Certamente, qualquer número de hifens no fonte permanece como estiverem em contextos literais, tais como `@code` e `@example`.
- Os caracteres de alimentação de formulário (*`CTRL-l`*) na entrada são manipulados conforme segue:

  PDF/DVI — Em texto normal, tratado como finalizante de qualquer parágrafo aberto; essencialmente ignorado entre parágrafos.

  Info — Saída como-é entre parágrafos (o uso mais comum deles); em outros contextos, eles podem ser tratados como espaços regulares (e então consolidados com espaço em branco envolvente).

  HTML — Escrito como uma entidade numérica, exceto contextos onde espaços são ignorados; por exemplo, em '`@footnote{ ^L foo}`', a alimentação de formulário é ignorada.

[1] A palavra argumento é oriunda da maneira como ela é empregada em matemática e não se refere a uma disputa entre duas pessoas; ela se refere a informação apresentada ao comando. De acordo com o *Oxford English Dictionary*, a palavra é derivada do Latim para *tornar claro, provar*; então, ela veio a significar 'a evidência oferecida como prova', o que é dizer 'a informação oferecida', o que leva ao seu significado matemático. Em sua outra derivação subjacente, a palavra veio a significar 'afirmar em um modo contra o qual outros podem fazer contra afirmações', o que levou ao significado de 'argumento' como uma disputa.

XML Mantenha-os em qualquer lugar; em atributos, encapsulados como '`\f`'; também, '\' é encapsulado como '`\\`' e uma nova linha como '`\n`'.

Docbook Completamente removidos, dado que não são permitidos.

Como você pode ver, por causa dessas exigências diferentes dos formatos de saída, não é possível se usar alimentações de formulário completamente portáveis.

- **Aviso:** Por último, não use caracteres de tabulação em um arquivo Texinfo! (Exceto, talvez, em modos textuais.) TEX usa fontes de largura variável, o que significa que é impraticável, na melhor das hipóteses, se definir uma tabulação para funcionar em todas as circunstâncias. Consequentemente, TEX trata tabulações como espaços únicos, e isso não é o que eles aparentam no fonte. Além disso, `makeinfo` não faz nada especial com tabulações, e então um caractere de tabulação em teu arquivo de entrada geralmente terá uma aparência diferente na saída.

  Para evitar esse problema, o modo Texinfo em GNU Emacs insere espaços múltiplos quando você pressiona a tecla `TAB`. Ainda, você pode executar `untabify` em Emacs para converter tabulações em uma região para espaços múltiplos, ou usar o comando `unexpand` a partir do shell.

2.2 Comentários

Você pode escrever comentários em um arquivo Texinfo usando o comando `@comment`, o qual pode ser abreviado para `@c`. Tais comentários são para uma pessoa examinando o arquivo fonte Texinfo. Todo o texto em uma linha que se seguir ou a `@comment` ou a `@c` é um comentário; o restante da linha não aparece na saída visível. (Para ser preciso, o caractere após o `@c` ou `@comment` deve necessariamente ser algo que não um traço ou alfanumérico, ou será tomado como sendo parte do comando.)

Frequentemente, você pode escrever o `@comment` ou `@c` no meio de uma linha, e somente o texto que se seguir após o comando `@comment` ou `@c` não aparece; porém alguns comandos, tais como `@settitle`, funcionam sobre uma linha inteira. Você não pode usar `@comment` ou `@c` dentro de uma linha que se inicie com um tal comando.

Em casos de invocações de comando aninhado, definições complicadas de macro, etc., `@c` e `@comment` podem provocar um erro quando do processamento com TEX. Por conseguinte, você também pode usar o caractere *`DEL`* (decimal 127 ASCII, hexadecimal 0x7f, octal 0177) como um verdadeiro caractere de comentário do TEX (código de categoria 14, nos internos do TEX). Tudo na linha após o *`DEL`* será ignorado.

Você também pode ter trechos longos de texto ignorados pelos processadores Texinfo com os comandos `@ignore` e `@end ignore`. Escreva cada um desses comandos em sua própria linha, iniciando cada comando no começo da linha. O texto entre esses dois comandos não aparece na saída processada. Você pode usar `@ignore` e `@end ignore` para escrever comentários. (Para algumas advertências relativamente ao aninhamento de tais comandos, veja-se Seção 16.7 [Aninhamento de Condicional], Página 136.)

2.3 O Que um Arquivo do Texinfo Deve Ter

Por convenção, o nome de um arquivo Texinfo termina com uma das extensões `.texinfo`, `.texi`, `.txi`, ou `.tex`.[2]

Para fazer com que seja um manual impresso e outros formatos de saída, um arquivo Texinfo deve necessariamente iniciar com linhas como estas:

[2] As extensões mais longas são as preferidas, dado que elas descrevem mais claramente a um leitor humano a natureza do arquivo. As extensões mais curtas são para sistemas operacionais que não conseguem lidar com nomes longos de arquivos.

```
\input texinfo
@settitle nome-do-manual
```

O conteúdo do arquivo segue esse início, e então você deve finalizar o fonte Texinfo com uma linha como esta:

```
@bye
```

Aqui está uma explanação:

- A linha '`\input texinfo`' manda TEX usar o arquivo `texinfo.tex`, o qual instrui TEX como traduzir os comandos @ Texinfo em comandos de tipografia TEX. (Note-se o uso da barra invertida, '\'; isso é correto para TEX.)
- A linha `@settitle` especifica um título para os cabeçalhos de página (ou rodapés) do manual impresso, e o título padrão e descrição de documento para o marcador '`<head>`' em HTML. Estritamente falando, `@settitle` é opcional—se você não se importar de ver o seu documento sendo intitulado 'Untitled'.
- A linha `@bye` ao final do arquivo em uma linha própria informa aos formatadores que o arquivo está finalizado e para parar de formatar. Se você deixar isso fora, você será colocado no prompt de TEX ao final da execução.

Além disso, você geralmente proverá um arquivo Texinfo com uma página de título, índices, e semelhantes, tudo isso é explanado neste manual. Porém, o mínimo, que pode ser útil para documentos curtos, é apenas as duas linhas no início e aquela no final.

2.4 Um Arquivo Curto de Amostra do Texinfo

Aqui está um curto, porém completo, arquivo Texinfo, de forma que você possa ver como um fonte Texinfo se aparenta na prática. As primeiras três partes do arquivo são em sua maioria clichê: quando da escrita de um manual, você simplesmente modifica os nomes conforme apropriado.

O arquivo completo, sem os comentários intercalados, é mostrado em Seção C.1 [Arquivo Curto de Amostra do Texinfo], Página 232.

Veja-se Capítulo 3 [Iniciando e Finalizando um Arquivo], Página 14, para mais documentação acerca dos comandos listados aqui.

Cabeçalho

O cabeçalho diz a TEX qual arquivo de definições usar, nomeia o manual, e realiza outras tais tarefas domésticas.

```
\input texinfo
@settitle Manual de Amostra 1.0
```

Descrição Resumida e Direitos Autorais

Este segmento descreve o documento e contém o aviso de direitos autorais e permissões de cópia. Isso é feito com o comando `@copying`.

Um manual real inclui mais texto aqui, de acordo com a licença sob a qual ele é distribuído. Veja-se Seção C.2 [Textos GNU de Amostra], Página 233.

```
@copying
Este é um exemplo curto de um arquivo completo Texinfo, versão 1.0.

Direitos autorais @copyright{} 2016 Free Software Foundation, Inc.
@end copying
```

Página de Título, Direitos Autorais, Conteúdo

O segmento de título e direitos autorais contém as páginas de título e de direitos autorais para o manual impresso. O segmento deve necessariamente estar incluso entre os comandos `@titlepage` e `@end titlepage`. A página de título e de direitos autorais não aparece na saída online.

Nós usamos o comando `@insertcopying` para incluir o texto de permissão a partir da seção anterior, em vez de escrevê-lo outra vez; o texto é colocado no verso da página de título. O comando `@contents` gera um sumário.

```
@titlepage
@title Título de Amostra

@c Os dois comandos seguintes iniciam a página de direitos autorais.
@page
@vskip 0pt plus 1filll
@insertcopying
@end titlepage

@c Coloca o sumário no início.
@contents
```

Nó 'Top' e Menu Mestre

O nó 'Top' começa a saída online; ela não aparece no manual impresso. Nós repetimos a descrição curta do início do texto '`@copying`', porém não há a necessidade de se repetir a informação de direitos autorais, de forma que nós não usamos '`@insertcopying`' aqui.

O próprio comando '`@top`' ajuda `makeinfo` a determinar o relacionamento entre os nós. O nó 'Top' contém, pelo menos, um *menu* de alto nível listando os capítulos, e possivelmente um *Menu Mestre* listando todos os nós no documento inteiro.

```
@ifnottex
@node Top
@top Amostra Curta

Este é um arquivo curto de amostra Texinfo.
@end ifnottex

@menu
* Primeiro Capítulo:: O primeiro capítulo é o único capítulo nesta
amostra.
* Índice::            Índice completo.
@end menu
```

O Corpo do Documento

O segmento corpo contém todo o texto do documento, porém não os índices ou sumário. Este exemplo ilustra um nó e um capítulo contendo uma lista enumerada.

```
@node Primeiro Capítulo
@chapter Primeiro Capítulo

@cindex capítulo, primeiro

Este é o primeiro capítulo.
@cindex entrada de índice, um outro
```

```
Aqui está uma lista numerada.

@enumerate
@item
Este é o primeiro item.

@item
Este é o segundo item.
@end enumerate
```

O Fim do Documento

Isto pode conter comandos para a impressão de índices, o fecha com o comando `@bye`, o qual marca o fim do documento.

```
@node Índice
@unnumbered Índice

@printindex cp

@bye
```

Alguns Resultados

Aqui está com o que se parece o conteúdo do primeiro capítulo da amostra:

Este é o primeiro capítulo.

Aqui está uma lista numerada.

1. Este é o primeiro item.
2. Este é o segundo item.

3 Começando e Terminando um Arquivo Texinfo

Este capítulo se estende sobre o arquivo fonte mínimo completo de Texinfo previamente dado (veja-se Seção 2.4 [Amostra Curta], Página 11).

Certos pedaços de informação devem necessariamente ser providos no início de um arquivo Texinfo, tais como o título do documento e o nó Top. Um sumário geralmente também é produzido aqui.

Texto direto fora de qualquer comando antes do nó Top deveria ser evitado. Tal texto é tratado diferentemente nos diferentes formatos de saída: no momento da escrita, é visível em TEX e HTML, por padrão não mostrado em leitores Info, e assim por diante.

3.1 Exemplo de Início do Arquivo do Texinfo

A amostra seguinte mostra o que é necessário. Os elementos dados aqui são explanados em maiores detalhes nas seções seguintes. Outros comandos frequentemente são incluídos no início de arquivos Texinfo, porém aqueles aqui são os mais críticos.

Veja-se Seção C.2 [Textos GNU de Amostra], Página 233, para os textos completos a serem usados em manuais GNU.

```
\input texinfo
@settitle nome-do-manual versão

@copying
Este manual é para programa, versão versão.

Direitos autorais @copyright{} anos titular-direitos-autorais.

@quotation
É concedida permissão para ...
@end quotation
@end copying

@titlepage
@title nome-do-manual-quando-impresso
@subtitle subtítulo-se-algum
@subtitle segundo-subtítulo
@author autor

@c  Os dois comandos seguintes
@c  iniciam a página de direitos autorais.
@page
@vskip 0pt plus 1filll
@insertcopying

Publicado por ...
@end titlepage

@c Então o sumário é impresso no início.
@contents

@ifnottex
@node Top
```

```
@top título

Este manual é para programa, versão versão.
@end ifnottex

@menu
* First Chapter::    Iniciando ...
* Second Chapter::       ... ...
* Copying::          Seus direitos e liberdades.
@end menu

@node Primeiro Capítulo
@chapter Primeiro Capítulo

@cindex primeiro capítulo
@cindex capítulo, primeiro
...
```

3.2 Cabeçalho do Arquivo do Texinfo

Os arquivos Texinfo iniciam com pelo menos duas linhas. Essas são a linha `\input texinfo` e a linha `@settitle`.

Também, se você desejar formatar somente parte do arquivo Texinfo em Emacs, você deve necessariamente escrever a linha `@settitle` entre as linhas 'start-of-header' e 'end-of-header'. Essas linhas 'start-' e 'end-of-header' são opcionais, porém elas não produzem danos, de forma que você pode muito bem sempre incluí-las.

Qualquer comando que afete a formatação do documento como um todo faz sentido incluir no cabeçalho. `@synindex` (veja-se Seção 11.5.2 [`@synindex`], Página 93), por exemplo, é outro comando frequentemente incluído no cabeçalho.

Assim, o início de um arquivo Texinfo se parece aproximadamente com isto:

```
\input texinfo
@settitle Manual de Amostra 1.0
```

(Veja-se Seção C.2 [Textos GNU de Amostra], Página 233, para textos de amostra completos).

3.2.1 A Primeira Linha de um Arquivo do Texinfo

Cada arquivo Texinfo que é para ser a entrada de alto nível para TEX deve necessariamente iniciar com uma linha que se parece com isto:

```
\input texinfo
```

Quando o arquivo é processado por TEX, o comando '`\input texinfo`' manda TEX carregar as macros necessárias para o processamento de um arquivo Texinfo. Essas estão em um arquivo chamado `texinfo.tex`, o qual deveria ter sido instalado em seu sistema juntamente com o software TEX ou Texinfo. TEX usa uma barra invertida, '\', para marcar o início de um comando, exatamente como Texinfo usa '`@`'. O arquivo `texinfo.tex` provoca a permuta de '\' para '`@`'; antes que a permuta ocorra, TEX exige '\', o que é o motivo pelo qual ela aparece no início do arquivo.

Você pode opcionalmente seguir essa linha com um comentário para informar a GNU Emacs para usar o modo Texinfo quando o arquivo for editado:

```
\input texinfo   @c -*-texinfo-*-
```

Isso pode ser útil quando Emacs não detectar automaticamente o tipo do arquivo a partir da extensão do arquivo.

3.2.2 Início de Cabeçalho

Uma linha 'start-of-header' é um comentário Texinfo que se parece com isto:

```
@c %**start of header
```

Escreva a linha 'start-of-header' na segunda linha de um arquivo Texinfo. Prossiga a linha 'start-of-header' com uma linha '`@settitle`' e, opcionalmente, com outros comandos que globalmente afetam a formatação do documento, tais como `@synindex` ou `@footnotestyle`; e então por uma linha 'end-of-header' (veja Seção 3.2.5 [Fim de Cabeçalho], Página 17).

As linhas 'start-' e 'end-of-header' te permitem formatar somente parte de um arquivo Texinfo para Info ou impressão. Veja Seção 21.1.3 [Comandos `texinfo-format`], Página 186.

A estranha sequência de caracteres, '`%**`', é para assegurar que nenhum outro comentário seja acidentalmente tomado por uma linha 'start-of-header'. Você pode modificá-la se você desejar, configurando as variáveis Emacs `tex-start-of-header` e/ou `tex-end-of-header`. Veja Seção 19.6 [Impressão no Modo Texinfo], Página 155.

3.2.3 `@setfilename`: Configura o Nome do Arquivo de Saída

A linha `@setfilename` especifica o nome do arquivo de saída a ser gerado. Quando presente, ele deveria ser o primeiro comando Texinfo (isto é, após '`\input texinfo`'). Escreva o comando `@setfilename` no início de uma linha e siga-o na mesma linha pelo nome do arquivo Info.

```
@setfilename nome-arquivo-info
```

O nome deve necessariamente ser diferente do nome do arquivo Texinfo. Existem duas convenções para a escolha do nome: você ou pode remover a extensão (tal como '`.texi`') inteiramente do nome do arquivo de entrada; ou (recomendado) substitui-lo com a extensão '`.info`'.

Quando uma linha `@setfilename` está presente, os processadores Texinfo ignoram tudo escrito antes da linha `@setfilename`. Isso é o motivo da primeiríssima linha do arquivo (a linha `\input`) não aparecer na saída.

Se não existir uma linha `@setfilename`, `makeinfo` usa o arquivo de entrada para determinar o nome de saída: primeiro, qualquer das extensões `.texi`, `.tex`, `.txi` ou `.texinfo` é removida do nome do arquivo de entrada; então, a extensão específica do formato de saída é adicionada—`.html` quando da geração de HTML; `.info` quando da geração de Info, etc. A linha `\input` ainda é ignorada nesse processamento, bem como as linhas em branco iniciais.

Quando da produção de outro formato de saída, `makeinfo` substituirá qualquer extensão final com a extensão de saída específica de formato ('`html`' quando da geração de HTML, por exemplo), ou adicionar um ponto seguido pela extensão ('`.html`' para HTML), se o nome dado não tiver extensão.

`@setfilename` costumava ser exigido pelos processadores Texinfo, e alguns outros programas ainda podem esperar que ele esteja presente; por exemplo, Automake (veja-se Seção "Texinfo" em *GNU Automake*).

Apesar que uma extensão '`.info`' explícita seja preferível, alguns sistemas operacionais não conseguem lidar com nomes longos de arquivo. Você pode acabar com problemas mesmo quando o nome de arquivo que você especificar for ele próprio curto o suficiente. Isso ocorre porque os formatadores Info dividem um arquivo longo Info em sub arquivos curtos indiretos, e os nomeia acrescentando ao final '`-1`', '`-2`', ..., '`-10`', '`-11`', e assim por diante, ao nome de arquivo original. (Veja-se Seção 21.1.5 [Arquivos de Etiqueta e de Divisão], Página 187.) O nome de subarquivo `texinfo.info-10`, por exemplo, é longo demais para sistemas antigos com um limite de 14 caracteres em nomes de arquivo; assim, o nome de arquivo Info para este documento é `texinfo` em vez de `texinfo.info`. Quando `makeinfo` está em execução em sistemas operacionais tais como MS-DOS que impõe limites severos sobre nomes de arquivo, ele pode remover alguns caracteres do nome original do arquivo para deixar espaço suficiente para o sufixo do subarquivo, assim produzindo arquivos chamados `texin-10`, `gcc.i12`, etc.

Veja-se também a opção `--output` em Seção 20.2 [Invocando `texi2any`], Página 163.

3.2.4 `@settitle`: Configura o Título do Documento

Um arquivo Texinfo deveria conter uma linha que se pareça com isto:

```
@settitle título
```

Escreva o comando `@settitle` no início de uma linha e siga-o na mesma pelo título. Não escreva nada mais na linha. O comando `@settitle` deveria preceder tudo o que gera saída atual. O melhor lugar para ele é logo após o comando `@setfilename` (descrito na seção anterior).

Esse comando informa a TEX o título a usar em um cabeçalho ou rodapé para saída de lado duplo, no caso de tais cabeçalhos serem impressos. Para mais sobre cabeçalhos para TEX, veja-se Seção 3.4.5 [Geração de Cabeçalho], Página 22.

No arquivo HTML produzido por `makeinfo`, *título* serve como o '`<título>`' do documento. Ele também se torna a descrição padrão do documento na parte '`<head>`' (veja Seção 3.7.1 [`@documentdescription`], Página 25).

Quando a página de título for usada na saída, o título no comando `@settitle` não afeta o título conforme ele aparece na página de título. Assim, os dois não precisam coincidir exatamente. Uma prática que se recomenda é incluir a versão ou número de edição do manual no título `@settitle`; na página de título, o número de versão geralmente aparece como um `@subtitle`, de forma que ele poderia ser omitido do `@title`. Veja-se Seção 3.4.1 [`@titlepage`], Página 19.

3.2.5 Fim de Cabeçalho

Siga as linhas de cabeçalho com uma linha end-of-header, que é um comentário Texinfo que se parece com isto:

```
@c %**end of header
```

Veja-se Seção 3.2.2 [Início de Cabeçalho], Página 16.

3.3 Permissões do Documento

O aviso de direitos autorais e permissões de cópia para um documento precisam aparecer em vários lugares nos vários formatos de saída de Texinfo. Portanto, Texinfo provê um comando (`@copying`) para declarar esse texto uma vez, e outro comando (`@insertcopying`) para inserir o texto em pontos apropriados.

Esta seção é sobre a licença do documento Texinfo. Se o documento for um manual de software, o software está tipicamente sob uma licença diferente—para GNU e muitos outros pacotes de software livre, o software é usualmente publicado sob a GNU GPL, e os manuais são publicados sob a GNU FDL. É de muita ajuda declarar a licença do software do manual, porém fornecer o texto completo da licença de software não é necessariamente exigido.

3.3.1 `@copying`: Declare as Permissões de Cópia

O comando `@copying` deveria ser dado cedissimo no documento; a localização recomendada é logo após o material de cabeçalho (veja Seção 3.2 [Cabeçalho do Arquivo do Texinfo], Página 15). Convencionalmente consiste de uma sentença ou duas sobre o que é o programa, identificação da própria documentação, a linha de direitos autorais legal, e as permissões de cópia. Aqui está um exemplo esqueletal:

```
@copying
Este documento é para programa (versão versão, atualizada
date), o qual ...

Direitos autorais @copyright{} anos titular-direitos-autorais.
```

```
@quotation
É concedida permissão para ...
@end quotation
@end copying
```

O `@quotation` não tem significado legal; ele está lá para melhorar a legibilidade em alguns contextos.

O texto de `@copying` é produzido como um comentário no início dos arquivos de saída Info, HTML, XML, e Docbook. O texto *não* é produzido implicitamente em texto plano ou TEX; depende de você usar `@insertcopying` para emitir a informação de direitos autorais. Veja a próxima seção para detalhes.

O comando `@copyright{}` gera um '`c`' dentro de um círculo quando o formato de saída suportar essa figura (imprimir e HTML sempre suportam, por exemplo). Quando a figura não for suportada na saída, o comando gera a sequência de três caracteres '`(C)`'.

O próprio aviso de direitos autorais tem a seguinte forma legalmente prescrita:

`Direitos autorais` © *`anos titular-direitos-autorais.`*

A palavra 'Copyright' deve necessariamente estar escrita em inglês, mesmo se o documento estiver escrito em outro idioma. Isso é devido ao direito internacional.

A lista dos anos deveria incluir todos os anos nos quais uma versão foi completada (mesmo se ela foi publicada em um ano subsequente). É mais simples que cada ano seja escrito individualmente e na íntegra, separados por vírgulas.

O titular (ou titulares) dos direitos autorais é quem quer que detenha os direitos autorais legais sobre o trabalho. No caso de trabalhos atribuídos à FSF, o titular é 'Free Software Foundation, Inc.'.

A 'linha' de direitos autorais atualmente pode ser dividida em linhas múltiplas, tanto no documento fonte quanto na saída. Isso acontece frequentemente para documentos com um histórico longo, tendo muitos anos diferentes de publicação. Se você usar várias linhas, não endente qualquer delas (ou qualquer outra coisa no bloco `@copying`) no arquivo fonte.

Veja-se Seção "Copyright Notices" em *GNU Maintainer Information*, para informação adicional. Veja-se Seção C.2 [Textos GNU de Amostra], Página 233, para o texto completo a ser usado em manuais GNU. Veja-se Apêndice H [Licença GNU de Documentação Livre], Página 265, para a própria licença sob a qual GNU e outros manuais livres são distribuídos.

3.3.2 `@insertcopying`: Incluir Texto de Permissões

O comando `@insertcopying` simplesmente é escrito em uma linha própria, como esta:

```
@insertcopying
```

Isso insere o texto previamente definido por `@copying`. Para atender às exigências legais, o texto deve necessariamente ser usado na página de direitos autorais no manual impresso (veja-se Seção 3.4.4 [Direitos Autorais], Página 21).

O próprio comando `@copying` faz com que o texto das permissões apareça em um arquivo Info *antes* do primeiro nó. O texto também é copiado no início de cada arquivo dividido de saída Info, como é legalmente necessário. Essa localização implica que um humano lendo o manual usando Info *não* vê esse texto (exceto quando da uso do comando avançado *g* * de Info), porém isso não importa para propósitos legais, pois o texto está presente.

Similarmente, o texto `@copying` é automaticamente incluído no início de cada arquivo de saída HTML, como um comentário HTML. De novo, esse texto não é visível (a menos que o(a) leitor(a) visualize o fonte HTML).

O texto de permissões definido por `@copying` também aparece automaticamente no início dos arquivos de saída XML e Docbook.

3.4 Páginas de Título e de Direitos Autorais

Em saída impressa, o nome e autor do manual são usualmente impressos em uma página de título. A informação de direitos autorais é geralmente impressa no verso de página de título.

As páginas de título e de direitos autorais aparecem em manuais impressos, porém não na maior parte dos outros formatos de saída. Por causa disso, é possível usar vários comandos de tipografia ligeiramente obscuros que não são para ser usados no texto principal. Adicionalmente, esta parte do início de um arquio Texinfo contém o texto das permissões de cópia que aparece no manual impresso.

3.4.1 `@titlepage`

Inicie o material para a página de título e seguinte à página de direitos autorais com `@titlepage` em uma linha própria e finalize com `@end titlepage` em uma linha própria.

O comando `@end titlepage` inicia uma página nova e liga a numeração de página (veja Seção 3.4.5 [Geração de Cabeçalho], Página 22). Todo o material que você deseja que apareça em páginas não numeradas deveria ser colocado entre os comandos `@titlepage` e `@end titlepage`. Você pode forçar que o sumário apareça ali com o comando `@setcontentsaftertitlepage` (veja Seção 3.5 [Conteúdo], Página 22).

Ao usar o comando `@page` você pode forçar uma quebra de página dentro da região delineada pelos comandos `@titlepage` e `@end titlepage` e assim criar mais que uma página não numerada. É assim que a página de direitos autorais é produzida. (O comando `@titlepage` talvez pudesse ter sido melhor nomeado de `@titleandadditionalpages`, porém isso poderia ter sido até certo ponto longo!)

Quando você escreve um manual acerca de um programa de computador, você deveria escrever, na página de título, a versão do programa para a qual o manual se aplica. Se o manual muda mais frequentemente que o programa ou é independente dele, você também deveria incluir um número de edição[1] para o manual. Isso ajuda os leitores a se manterem informados sobre qual manual é para qual versão do programa. (O nó 'Top' também deveria conter essa informação; veja-se Seção 3.6 [O Nó Top], Página 23).

Texinfo provê dois métodos principais para criar uma página de título. Um método usa os comandos `@titlefont`, `@sp`, e `@center` para gerar uma página de título na qual as palavras na página são centralizadas.

O segundo método usa os comandos `@title`, `@subtitle`, e `@author` para criar uma página de título com traços negros sob as linhas de título e autor e o texto de subtítulo colocado rente ao lado direito da página. Com esse método, você não especifica nada da formatação atual da página de título. Você especifica o texto que você deseja, e Texinfo faz a formatação.

Você pode usar qualquer método, ou você pode combiná-los; veja os exemplos nas seções abaixo.

Para documentos suficientemente simples, e para a página de anterrosto em formatador tradicional de livro, Texinfo também provê um comando, `@shorttitlepage`, o qual toma o restante da linha como o título. O argumento é tipografado em uma página própria e seguido por uma página em branco.

3.4.2 `@titlefont`, `@center` e `@sp`

Você pode usar os comandos `@titlefont`, `@sp`, e `@center` para criar uma página de título para um documento impresso. (Esse é o primeiro de dois métodos para a criação de uma página de título em Texinfo).

[1] Entendeu-se que é útil se referir a versões de manuais independentes como 'edições' e versões de programas como 'versões'; do contrário, nós achamos que estamos sujeitos a confundir cada um em conversas, referenciando a ambos, a documentação e o software, com as mesmas palavras.

Use o comando `@titlefont` para selecionar uma fonte larga adequada para o próprio título. Você pode usar `@titlefont` mais que uma vez se você tiver um título especialmente longo.

Para saída HTML, cada comando `@titlefont` produz um cabeçalho `<h1>`, porém o `<title>` do documento HTML não é afetado. Para isso, você deve necessariamente colocar um comando `@settitle` antes do comando `@titlefont` (veja-se Seção 3.2.4 [`@settitle`], Página 17).

Por exemplo:

```
@titlefont{Texinfo}
```

Use o comando `@center` no início de uma linha para centralizar o restante do texto naquela linha. Assim,

```
@center @titlefont{Texinfo}
```

centraliza o título, o qual neste exemplo é "Texinfo" impresso na fonte de título.

Use o comando `@sp` para inserir espaço vertical. Por exemplo:

```
@sp 2
```

Isso insere duas linhas em branco na página impressa. (Veja-se Seção 13.7 [`@sp`], Página 112, para mais informação acerca do comando `@sp`).

Um modelo para esse método se parece com isto:

```
@titlepage
@sp 10
@center @titlefont{nome-do-manual-quando-impresso}
@sp 2
@center subtítulo-se-algum
@sp 2
@center autor
...
@end titlepage
```

O espaçamento do exemplo se encaixa em um manual de 8.5 por 11 polegadas.

Você de fato pode usar esses comandos em qualquer lugar, não somente em uma página de título, porém dado que eles não são comandos lógicos de marcação, não se recomenda-os.

3.4.3 `@title`, `@subtitle`, e `@author`

Você pode usar os comandos `@title`, `@subtitle`, e `@author` para criar uma página de título na qual o espaçamento vertical e horizontal seja feito para você automaticamente. Isso contrasta com o método descrito na seção anterior, no qual o comando `@sp` é necessário para ajustar o espaçamento vertical.

Escreva os comandos `@title`, `@subtitle`, ou `@author` no início de uma linha seguido pelo título, subtítulo, ou autor. O comando `@author` pode ser usado para uma citação em um bloco `@quotation` (veja-se Seção 8.2 [`@quotation`], Página 67); exceto para isso, é um erro usar quaisquer desses comandos fora de `@titlepage`.

O comando `@title` produz uma linha na qual o título é configurado rente ao lado esquerdo da página em um fonte mais larga que a normal. O título é sublinhado com uma linha preta. O título deve necessariamente ser dado em uma linha única no arquivo fonte; ele será quebrado em múltiplas linhas de saída se necessário.

Para títulos longos, o comando `@*` pode ser usado para especificar as quebras de linha em títulos longos se as quebras automáticas não servirem. Tais quebras explícitas de linha geralmente são refletidas em todos os formatos de saída; se você desejar somente especificá-las para a saída impressa, use um condicional (veja-se Capítulo 16 [Condicionais], Página 128). Por exemplo:

```
@title Este Título Longo@inlinefmt{tex,@*} É Quebrado em @TeX{}
```

O comando `@subtitle` configura subtítulos em uma fonte de tamanho normal rente ao lado direito da página.

O comando `@author` configura os nomes do autor ou autores em uma fonte de tamanho médio rente ao lado esquerdo da página em uma linha próxima ao pé da página de título. Os nomes são seguidos por uma linha preta que é mais fina que a linha que sublinha o título.

Existem duas maneiras de se usar o comando `@author`: você pode escrever o nome ou nomes na parte restante da linha que inicia com um comando `@author`:

```
@author por Jane Smith e John Doe
```

ou você pode escrever os nomes um acima do outro, usando múltiplos comandos `@author`:

```
@author Jane Smith
@author John Doe
```

Um modelo para esse método se parece com isto:

```
@titlepage
@title nome-do-manual-quando-impresso
@subtitle subtítulo-se-algum
@subtitle segundo-subtítulo
@author autor
@page
...
@end titlepage
```

3.4.4 Página de Direitos Autorais

Por acordo internacional, o aviso de direitos autorais para um livro deve necessariamente estar ou na página de título ou no verso da página de título. Quando o aviso de direitos autorais estiver no verso da página de título, essa página é costumeiramente não numerada. Portanto, em Texinfo, a informação na página de direitos autorais deveria estar entre os comandos `@titlepage` e `@end titlepage`.

Use o comando `@page` para provocar uma quebra de página. Para empurrar o aviso de direitos autorais e o outro texto na página de direitos autorais em direção ao pé da página, use o seguinte encantamento após `@page`:

```
@vskip 0pt plus 1filll
```

O comando `@vskip` insere espaço em branco na saída TEX; ele é ignorado em todos os outros formatos de saída. O '`0pt plus 1filll`' significa colocar em zero pontos de espaço em branco obrigatório, e tantos espaços em branco opcionais quantos sejam necessários para empurrar o texto seguinte para o pé da página. Note o uso de três '`l`'s na palavra '`filll`'; isso está correto.

Para inserir o próprio texto de direitos autorais, escreva `@insertcopying` depois (veja-se Seção 3.3 [Permissões do Documento], Página 17):

```
@insertcopying
```

Siga o texto de direitos autorais pelo editor, números ISBN, créditos de arte de capa, e outras tais informações.

Aqui está um exemplo de como colocar tudo isso junto:

```
@titlepage
...
@page
@vskip 0pt plus 1filll
@insertcopying

Publicado por ...
```

```
Arte de capa por ...
@end titlepage
```

Nós temos um caso especial a considerar: para saída de texto plano, você deve necessariamente inserir a informação de direitos autorais explicitamente se você deseja que ela apareça. Por exemplo, você poderia ter o seguinte após a página de direitos autorais:

```
@ifplaintext
@insertcopying
@end ifplaintext
```

Você poderia incluir outras informações de título para a saída de texto plano no mesmo lugar.

3.4.5 Geração de Cabeçalho

Como todos os comandos `@end` (veja-se Capítulo 8 [Citações e Exemplos], Página 66), o comando `@end titlepage` deve estar escrito no início de uma linha própria, com somente um espaço entre o `@end` e o `titlepage`. Ele somente marca o fim das páginas de título e de direitos autorais, porém também manda que TEX inicie a geração de cabeçalhos de página e de números de página.

Texinfo tem dois formatos padrão de cabeçalho de página, um para documentos impressos em um lado de cada folha de papel (impressão de lado único), e o outro para documentos impressos em ambos os lados de cada folha (impressão de lado duplo).

Em generalidade plena, você pode controlar os cabeçalhos em diferentes maneiras:

- A maneira convencional é escrever um comando `@setchapternewpage` antes dos comandos de página de título, se exigido, e então ter o comando `@end titlepage` iniciando a geração de cabeçalhos de página na maneira desejada.

  A maioria dos documentos é formatada com os cabeçalhos padrão lado único ou lado duplo, (as vezes) usando `@setchapternewpageodd` para impressão em lado duplo e (quase sempre) sem comando `@setchapternewpage` para impressão de lado único (veja-se Seção 3.7.2 [`@setchapternewpage`], Página 25).
- Alternativamente, você pode usar o comando `@headings` para prevenir que os cabeçalhos de página sejam gerados ou para iniciá-los para, ou impressão de lado único, ou duplo. Escreva um comando `@headings` imediatamente após o comando `@end titlepage`. Para desligar os cabeçalhos, escreva `@headings off`. Veja-se Seção 3.7.3 [`@headings`], Página 26.
- Ou, você pode especificar o seu próprio formato de cabeçalho e rodapé de página. Veja Apêndice E [Cabeçalhos], Página 248.

3.5 Gerando Um Sumário

`@chapter`, `@section`, e outros comandos de estruturamento (veja-se Capítulo 5 [Estruturamento de Capítulo], Página 39) fornecem a informação para produzir um Sumário, porém eles não fazem com que uma tabela atual apareça no manual. Para fazer isso, você deve necessariamente usar os comandos `@contents` e/ou `@summarycontents`.

`@contents`

Gera um Sumário em um manual impresso, incluindo todos os capítulos, seções, subseções, etc., bem como apêndices e capítulos não numerados. Os cabeçalhos gerados por `@majorheading`, `@chapheading`, e outros comandos `@...heading` não aparecem no Sumário (veja-se Seção 5.2 [Tipos de Comandos Estruturantes], Página 39).

`@shortcontents`
`@summarycontents`

(`@summarycontents` é um sinônimo para `@shortcontents`).

> Gera uma tabela curta ou sumário do conteúdo que lista somente os capítulos, apêndices e capítulos não numerados. Seções, subseções e subsubseções são omitidas. Somente um manual longo precisa de uma tabela curta de conteúdo em adição à tabela completa de conteúdo.

Ambos os comandos de conteúdo deveriam ser escritos em uma linha para cada um deles, e colocados próximo do início do arquivo, após o `@endtitlepage` (veja-se Seção 3.4.1 [`@titlepage`], Página 19), antes de qualquer comando de seccionamento. Os comandos de conteúdo automaticamente geram um cabeçalho estilo capítulo no topo da primeira página do Sumário, portanto não inclua qualquer comando de seccionamento tal qual `@unnumbered` antes deles.

Dado que um arquivo Info usa menus em vez de tabelas de conteúdo, os comandos de formatação de Info ignoram os comandos de conteúdo. Porém, o conteúdo é incluído na saída de texto plano (gerada por `makeinfo --plaintext`) e em outros formatos de saída, tais como HTML.

Quando `makeinfo` escreve uma tabela curta de conteúdo quando da produção de saída HTML, os links na tabela curta de conteúdo apontam para as entradas correspondentes na tabela completa de conteúdo, em vez de apontar para o texto do documento. Os links na tabela completa de conteúdo apontam para o texto principal do documento.

No passado, os comandos de conteúdo algumas vezes foram colocados ao final do arquivo, após quaisquer índices e pouco antes de `@bye`, porém não mais se recomenda isso.

Entretanto, dado que muitos documentos Texinfo existentes ainda tem o `@contents` ao final do manual, se você for um usuário imprimindo um manual, você pode desejar forçar que o conteúdo seja impresso após a página de título. Você pode fazer isso especificando `@setcontentsaftertitlepage` e/ou `@setshortcontentsaftertitlepage`. O primeiro imprime somente o conteúdo principal após o `@end titlepage`; o segundo imprime ambos o conteúdo curto e o conteúdo principal. Em qualquer caso, quaisquer `@contents` ou `@shortcontents` subsequentes são ignorados.

Você precisa incluir os comandos `@set...contentsaftertitlepage` logo no documento (logo após `@setfilename`, por exemplo). Recomenda-se usar `texi2dvi` (veja-se Seção 19.2 [Formatar com `texi2dvi`], Página 150) para especificar isso sem nenhuma alteração do arquivo fonte. Por exemplo:

```
texi2dvi --texinfo=@setcontentsaftertitlepage foo.texi
```

Uma invocação alternativa, usando `texi2any`:

```
texi2any --dvi --Xopt --texinfo=@setcontentsaftertitlepage foo.texi
```

3.6 O Nó 'Top' e Menu Mestre

O nó 'Top' é o nó no qual um leitor acessa um manual Info. Como tal, ele deveria iniciar com uma descrição breve do manual (incluindo o número de versão), e terminar com um menu mestre para o manual inteiro. Certamente você deveria incluir qualquer outra informação geral que você sinta que um leitor acharia de muita ajuda.

É convencional e desejável escrever uma linha de comando de seccionamento `@top` contendo o título do documento imediatamente após a linha `@node Top` (veja-se Seção 4.6 [Comando `@top`], Página 33).

O conteúdo do nó 'Top' deveria aparecer somente na saída online; nada dele deveria aparecer em saída impressa, então envolva-o entre os comandos `@ifnottex` e `@end ifnottex`. (TEX não imprime nem uma linha `@node` nem um menu; eles aparecem somente em Info; estritamente falando, você não está obrigado a envolver essas partes entre `@ifnottex` e `@end ifnottex`, porém é mais simples fazer isso. Veja-se Capítulo 16 [Conditionally Visible Text], Página 128).

3.6.1 Exemplo do Nó Top

Aqui está um exemplo de um nó Top.

```
@ifnottex
@node Top
@top Título de Amostra

Este é o texto do nó top.
@end ifnottex

Informação geral adicional.

@menu
* Primeiro Capítulo::
* Segundo Capítulo::
...
* Index::
@end menu
```

3.6.2 Partes de um Menu Mestre

Um *menu mestre* é o menu principal. É costumeiro incluir um menu detalhado listando todos os nós do documento nesse menu.

Como qualquer outro menu, um menu mestre é envolvido em `@menu` e `@end menu` e não aparece na saída impressa.

Geralmente, um menu mestre é dividido em partes.

- A primeira parte contém os nós principais no arquivo Texinfo: os nós para os capítulos, seções em forma de capítulo, e os apêndices.
- A segunda parte contém nós para os índices.
- A terceira e subsequentes partes contém uma listagem dos outros, nós de baixo nível, frequentemente ordenados por capítulo. Essa maneira, em vez de se ir ao longo de um menu intermediário, um investigador pode ir diretamente a um nó particular quando da busca por informação específica. Esses itens de menu não são exigidos; adicione-os se você pensar que eles são uma conveniência. Se você usá-los, coloque `@detailmenu` antes do primeiro, e `@end detailmenu` após o último; do contrário, `makeinfo` ficará confuso.

Cada seção no menu pode ser introduzida por uma linha descritiva. Contanto que a linha não se inicie com um asterisco, ela não será tratada como uma entrada de menu. (Veja-se Seção 4.9.1 [Escrevendo um Menu], Página 36, para mais informação).

Por exemplo, o menu mestre para este manual se parece com o seguinte (porém tem muito mais entradas):

```
@menu
* Condições de Cópia::  Seus direitos.
* Visão Geral::         Texinfo em resumo.
...
* Índices de Comandos e Variáveis::
* Índices Gerais::
```

```
@detailmenu
--- A Listagem Detalhada do Nó ---

Visão geral de Texinfo

* Relatando Bugs:: ...
...

Iniciando um Arquivo Texinfo

* Início de Amostra:: ...
...
@end detailmenu
@end menu
```

3.7 Comandos Globais de Documento

Ao lado dos comandos básicos mencionados nas seções anteriores, aqui estão comandos adicionais que afetam o documento como um todo. Eles geralmente são todos dados antes do nó Top, se forem dados afinal.

3.7.1 `@documentdescription`: Texto de Resumo

Quando da produção de saída HTML para um documento, `makeinfo` escreve um elemento '`<meta>`' no '`<head>`' para dar alguma ideia do conteúdo do documento. Por padrão, essa *descrição* é o título do documento, tomado a partir do comando `@settitle` (veja-se Seção 3.2.4 [`@settitle`], Página 17). Para mudar isso, use o ambiente `@documentdescription`, como em:

```
@documentdescription
texto descritivo.
@end documentdescription
```

Isso produzirá a saída seguinte no '`<head>`' do HTML:

```
<meta name=description content="texto descritivo.">
```

`@documentdescription` deve necessariamente estar especificado antes do primeiro nó do documento.

3.7.2 `@setchapternewpage`: Páginas em Branco Antes dos Capítulos

Em um oficialmente limitado livro, o texto normalmente é impresso em ambos os lados do papel, capítulos iniciam em páginas do lado direito, e as páginas do lado direito tem números ímpares. Porém em relatórios curtos, o texto frequentemente é impresso somente em um lado do papel. Também em relatórios curtos, capítulos as vezes não iniciam em páginas novas, mas são impressos na mesma página que o final do capítulo precedente, após uma quantidade pequena de espaço em branco vertical.

Você pode usar o comando `@setchapternewpage` com vários argumentos para especificar como TEX deveria iniciar capítulos e quando deveria formatar cabeçalhos para impressão em um ou ambos os lados do papel (impressão em lado único ou em lado duplo).

Escreva o comando `@setchapternewpage` no início de uma linha seguido pelo seu argumento.

Por exemplo, você escreveria o seguinte para fazer com que cada capítulo iniciasse em uma nova página com numeração ímpar:

```
@setchapternewpage odd
```

Você pode especificar uma de três alternativas com o comando `@setchapternewpage`:

`@setchapternewpage off`
: Provoca TEX a tipografar um novo capítulo na mesma página como o último capítulo, após pular algum espaço em branco vertical. Também faz com que TEX formate os cabeçalhos de página para impressão em lado único.

`@setchapternewpage on`
: Provoca TEX a iniciar novos capítulos em novas páginas e formatar os cabeçalhos de página para impressão em lado único. Essa é a forma frequentemente mais usada para relatórios curtos ou impressão pessoal. Essa é a padrão.

`@setchapternewpage odd`
: Provoca TEX a iniciar novos capítulos em novas, páginas de número ímpar (páginas de lado direito) e a tipografar para impressão em lado duplo. Essa é a forma frequentemente mais usada para livros e manuais.

Texinfo não tem um comando `@setchapternewpage even`, pois não há tradição de impressão de capítulos de início ou livros em uma página de número par.

Se você não gosta dos cabeçalhos padrão que `@setchapternewpage` configura, você pode explicitamente controlá-los com o comando `@headings`. Veja-se Seção 3.7.3 [`@headings`], Página 26.

No início de um manual ou livro, as páginas não numeradas—por exemplo, as páginas de título e de direitos autorais de um livro não são numeradas. Por convenção, as páginas de sumário e frontal são numeradas com numerais romanos e não em sequência com o restante do documento.

O `@setchapternewpage` não tem efeito em formatos de saída que não tem páginas, tais como Info e HTML.

Recomenda-se não incluir qualquer comando `@setchapternewpage` em seu fonte de documento de jeito nenhum, dado que tal paginação desejada não é intrínseca ao documento. Para uma execução particular de impressão em papel, se você não deseja que a saída padrão (sem páginas em branco, mesmos cabeçalhos em todas as páginas) use a opção `--texinfo` a `texi2dvi` para especificar a saída que você deseja.

3.7.3 O Comando `@headings`

O comando `@headings` raramente é usado. Ele especifica o tipo de cabeçalhos e rodapés a imprimir em cada página. Usualmente, isso é controlado pelo comando `@setchapternewpage`. Você precisa do comando `@headings` somente se o comando `@setchapternewpage` não fizer o que você desejar, ou se você desejar desligar os cabeçalhos de página pré-definidos antes de definir os seus próprios. Escreva um comando `@headings` imediatamente após o comando `@end titlepage`.

Você pode usar `@headings` conforme a seguir:

`@headings off`
: Desliga a impressão de cabeçalhos de páginas.

`@headings single`
: Liga os cabeçalhos de página apropriados para a impressão em lado único.

`@headings double`
: Liga os cabeçalhos de página apropriados para a impressão em lado duplo.

`@headings singleafter`
`@headings doubleafter`
: Liga os cabeçalhos `single` ou `double`, respectivamente, após ser produzida a saída da página atual.

`@headings on`
: Liga os cabeçalhos de página: `single` se '`@setchapternewpageon`'; e `double` do contrário.

Por exemplo, suponha que você escreva `@setchapternewpage off` antes do comando `@titlepage` para informar a TEX para iniciar um capítulo novo na mesma página como o final do mais recente capítulo. Esse comando também faz com que TEX tipografe os cabeçalhos de página para a impressão em lado único. Para fazer com que TEX tipografe para impressão em lado duplo, escreva `@headingsdouble` após o comando `@end titlepage`.

Você pode fazer com que TEX pare de gerar quaisquer cabeçalhos de página em quaisquer circunstâncias, escrevendo `@headings off` em uma linha própria imediatamente após a linha contendo o comando `@end titlepage`, como isto:

```
@end titlepage
@headings off
```

O comando `@headings off` anula o comando `@end titlepage`, o qual do contrário faz com que TEX imprima os cabeçalhos de página.

Você também pode especificar seu próprio estilo de cabeçalho e de rodapé de página. Veja-se Apêndice E [Cabeçalhos de Página], Página 248, para mais informação.

3.7.4 `@paragraphindent`: Controlando o Recuo de Parágrafo

Os processadores de Texinfo podem inserir espaços em branco no início da primeira linha de cada parágrafo, consequentemente recuando esse parágrafo. Você pode usar o comando `@paragraphindent` para especificar tal recuo. Escreva um comando `@paragraphindent` no início de uma linha seguido por ou '`asis`' ou um número:

```
@paragraphindent indent
```

O recuo ocorre de acordo com o valor de *indent*:

`asis`
: Não modifica o recuo existente (não implementado em TEX).

`none`
0
: Omite todos os recuos.

n
: Recua *n* caracteres de espaço em saída Info, *n* em TEX.

O valor padrão de *indent* é três (3). `@paragraphindent` é ignorado para saída HTML.

É melhor escrever o comando `@paragraphindent` antes da linha de fim de cabeçalho, no início de um arquivo Texinfo, de maneira que os comandos de formatação de região recuem os parágrafos conforme especificado. Veja-se Seção 3.2.2 [Início de Cabeçalho], Página 16.

3.7.5 `@firstparagraphindent`: Recuando Após Cabeçalhos

Conforme você pode ver no presente manual, o primeiro parágrafo em qualquer seção não é recuado por padrão. Tipograficamente, o recuo é um separador de parágrafos, o que significa que ele é desnecessário quando uma nova seção se inicia. Esse recuo é controlado com o comando `@firstparagraphindent`:

```
@firstparagraphindent word
```

O primeiro parágrafo após um cabeçalho é recuado de acordo com o valor de *word*:

`none`
: Evita que o primeiro parágrafo seja recuado (padrão). Essa opção é ignorada por `makeinfo` se `@paragraphindent asis` estiver em efeito.

`insert`
: Inclui o recuo normal de parágrafo. Isso respeita o recuo de parágrafo configurado por um comando `@paragraphindent` (veja-se Seção 3.7.4 [`@paragraphindent`], Página 27).

`@firstparagraphindent` é ignorado para saída HTML e Docbook.

É melhor escrever o comando `@firstparagraphindent` antes da linha de fim de cabeçalho, no início de um arquivo Texinfo, de maneira que os comandos de formatação de região recuem os parágrafos conforme especificado. Veja-se Seção 3.2.2 [Início de Cabeçalho], Página 16.

3.7.6 `@exampleindent`: Recuo de Ambiente

Os processadores Texinfo recuam cada linha de `@example` e ambientes similares. Você pode usar o comando `@exampleindent` para especificar tal recuo. Escreva um comando `@exampleindent` no início de uma linha seguido por ou '`asis`' ou um número:

```
@exampleindent indent
```

O recuo é de acordo com o valor de *indent*:

`asis` Não modifica o recuo existente (não implementado em TEX).

0 Omite todos os recuos.

n Recua ambientes em *n* caracteres de espaço em saída Info, *n* em TEX.

O valor padrão de *indent* é cinco (5) espaços em Info, e 0.4 in em TEX, o que é de alguma maneira menos. (A redução é para ajudar TEX a encaixar mais caracteres em linhas físicas).

É melhor escrever o comando `@exampleindent` antes da linha de fim de cabeçalho, no início de um arquivo Texinfo, de maneira que os comandos de formatação de região recuem os parágrafos conforme especificado. Veja Seção 3.2.2 [Início de Cabeçalho], Página 16.

3.8 Finalizando um Arquivo do Texinfo

O final de um arquivo Texinfo deveria incluir comandos para criar índices (veja-se Seção 11.4 [Imprimindo Índices e Menus], Página 91), e o comando `@bye` para marcar a última linha a ser processada. Por exemplo:

```
@node Index
@unnumbered Index

@printindex cp

@bye
```

Um comando `@bye` termina o processamento Texinfo. Nenhum dos formatadores processa nada seguinte a `@bye`; qualquer de tal texto é completamente ignorado. O comando `@bye` deveria estar em uma linha própria.

Assim, se você desejar, você pode seguir a linha `@bye` com notas arbitrárias. Também, você pode seguir a linha `@bye` com uma lista de variáveis locais para Emacs, a maioria das vezes tipicamente um '`compile-command`' (veja-se Seção 19.7 [Usando a Lista de Variáveis Locais], Página 156).

4 Nós

Um *nó* é uma região de texto que inicia no comando `@node`, e continua até o próximo comando `@node`. Para especificar um nó, escreva um comando `@node` no início de uma linha, e siga-o com o nome do nó. Cada nó contém a discussão de um tópico. Os leitores Info exibem um nó por vez, e disponibilizam comandos para o usuário se movimentar a nós relacionados. A saída HTML pode ser navegada similarmente.

Os nós são usados como os alvos de referências cruzadas. As referências cruzadas, tais quais aquelas ao final desta sentença, são feitas com `@xref` e comandos relacionados; veja-se Capítulo 6 [Referências Cruzadas], Página 45. As referências cruzadas podem ser colocadas em qualquer parte do texto, e proporcionam uma maneira de representar links que não se encaixam em uma estrutura hierárquica.

Normalmente, você coloca um comando nó imediatamente antes de cada comando de estruturação de capítulo—por exemplo, uma linha `@section` ou `@subsection`. (Veja-se Capítulo 5 [Estruturamento de Capítulo], Página 39). Você deve necessariamente fazer isso, ainda que você não pretenda formatar o arquivo para Info. Isso é por que TEX usa ambos, nomes `@node` e nomes de estruturação de capítulo, na saída para referências cruzadas. A única vez que você similarmente usa os comandos de estruturação de capítulo sem também usar nós é se você estiver escrevendo um documento que não contenha referências cruzadas e somente será impresso, não transformado em Info, HTML, ou outros formatos.

4.1 Estrutura do Documento Texinfo

Os nós podem conter *menus*, os quais contém os nomes de *nós descendentes* dentro do nó pai; por exemplo, um nó correspondente a um capítulo poderia ter um menu de seções nesse capítulo. Os menus permitem ao usuário se movimentar aos nós filhos em uma maneira natural na saída em linha.

Adicionalmente, os nós contém *ponteiros de nó* que nomeiam outros nós. Os ponteiros 'Próximo' e 'Anterior' formam nós no mesmo nível de seccionamento em uma cadeia. Conforme você pode imaginar, o ponteiro 'Próximo' vincula ao próximo nó, e o ponteiro 'Anterior' vincula ao nó anterior. Assim, por exemplo, todos os nós que estão ao nível de seções dentro de um capítulo, são vinculados juntos, e a ordem dentro dessa cadeia é a mesma que a ordem dos filhos no menu do capítulo pai. Cada nó filho grava o nome de nó pai como o seu ponteiro 'Acima'.

A saída Info e HTML originada de `makeinfo` para cada nó inclui vínculos aos nós 'Próximo', 'Anterior', e 'Acima'. O HTML também usa o atributo `accesskey` com os valores '`n`', '`p`', e '`u`', respectivamente. Isso permite a pessoas que usem navegadores web a seguir a navegação usando (tipicamente) *`M-letter`*, por exemplo, *`M-n`* para o nó 'Próximo', a partir de qualquer lugar dentro do nó. Os ponteiros e menus de nó proveem estrutura para os arquivos Info, exatamente como capítulos, seções, subseções e semelhantes proveem estrutura para livros impressos. As duas estruturas teoricamente são distintas; na prática, entretanto, a estrutura de árvore de livros impressos é também essencialmente sempre usada para o nó e estrutura de menu, pois isso conduz a um documento que é mais fácil de seguir. Veja-se Seção 4.1 [Estrutura do Documento Texinfo], Página 29.

Tipicamente, a estrutura de seccionamento e a estrutura de nó são completamente paralelas, com um nó para cada capítulo, seção, etc., e com os nós seguindo o mesmo arranjo hierárquico que o seccionamento. Assim, se o nó estiver ao nível lógico de um capítulo, então seus nós filhos estão ao nível de seções; similarmente, os nós filhos de seções estão ao nível de subseções.

Apesar que é tecnicamente possível se criar documentos Texinfo com somente uma estrutura ou a outra, ou para as duas estruturas não estarem em paralelo, ou para ou o seccionamento ou estrutura de nó estarem anormalmente formadas, etc., isso *terminantemente não é recomendado.*

Conforme é do nosso conhecimento, todos os manuais Texinfo atualmente em uso geral seguem a estrutura paralela convencional.

4.2 Escolhendo Nomes de Nó

O nome de um nó identifica o nó. Para todos os detalhes de nomes de nó, veja-se Seção 4.4 [Exigências de Linha de Nó], Página 31).

Aqui estão algumas sugestões para nomes de nó:

- Tente pegar nomes de nó que sejam informativos, porém curtos.

  No arquivo Info, o nome de arquivo, nome de nó, e nomes de ponteiro são todos inseridos em uma linha, os quais podem invadir a borda direita da janela. (Isso não causa um problema com Info, porém é horrível).
- Tente pegar nomes de nó que difiram uns dos outros próximo do início de seus nomes. Dessa maneira, é fácil usar a complementação automática de nome em Info.
- Convencionalmente, os nomes de nó são capitalizados da mesma maneira que os títulos de seção e capítulo. Neste manual, as palavras iniciais e significantes são capitalizadas; outras não o são. Em outros manuais, somente as palavras iniciais e substantivos próprios são capitalizados. Qualquer forma está boa; nós somente recomendamos ser consistente.

Por que os nomes de nó são usados em referências cruzadas, não é desejável casualmente modificá-los uma vez publicados. Tais mudanças de nome invalidam as referências a partir de outros manuais, a partir de arquivamentos de correspondências, e assim por diante. Veja-se Seção 22.4.7 [Preservação de Link Xref do HTML], Página 203.

Os ponteiros a partir de um dado nó te habilitam a alcançar outros nós e consistem simplesmente dos nomes de tais nós. Os ponteiros usualmente não estão especificados explicitamente, conforme `makeinfo` possa determiná-los (veja-se Seção 4.8 [Criação de Ponteiros do `makeinfo`], Página 35).

Normalmente, um ponteiro 'Acima' de nó contém o nome do nó cujo menu menciona aquele nó. O ponteiro 'Próximo' de nó contém o nome do nó que segue o presente nó naquele menu e seu ponteiro 'Anterior' contém o nome do nó que o precede naquele menu. Quando um nó 'Anterior' de nó for o mesmo que seu nó 'Acima', ambos os ponteiros nomeiam o mesmo nó.

Usualmente, o primeiro nó de um arquio Texinfo é o nó 'Top', e o seu ponteiro 'Acima' aponta para o arquivo `dir`, o qual contém o menu principal para todos os Info.

4.3 Escrevendo uma Linha de `@node`

A maneira mais fácil de se escrever uma linha `@node` é escrever `@node` no início de uma linha e então o nome do nó, como isto:

```
@node nome_do_nodo
```

Após se ter inserido uma linha `@node`, você deveria imediatamente escrever um comando @ para o capítulo ou seção e inserir o nome desse capítulo ou seção. Próximo (e isto é importante!), coloque várias entradas de índices. Usualmente, você encontrará pelo menos duas e frequentemente por volta de quatro ou cinco maneiras de se referir ao nó no índice. Use-as todas. Isso tornará muito mais fácil para as pessoas encontrar o nó.

Se você desejar, você pode ignorar as linhas `@node` completamente em seu primeiro rascunho e então usar o comando `texinfo-insert-node-lines` para criar as linhas `@node` para você. Entretanto, não se recomenda essa prática. É melhor nomear o nó ao mesmo tempo em que você escreve um segmento, de forma que você possa facilmente produzir referências cruzadas. As referências cruzadas úteis são uma característica especialmente importante de um bom manual Texinfo.

Mesmo quando você explicitamente especificar todos os ponteiros, você não pode escrever os nós no arquivo fonte Texinfo em uma ordem arbitrária! Porque os formatadores devem necessariamente processar o arquivo sequencialmente, sem considerar os ponteiros de nó, você deve necessariamente escrever os nós na ordem que você os desejar para aparecer na saída. Para o formato Info, alguém pode imaginar que a ordem pode não importar, porém ela importa para os outros formatos.

Você pode opcionalmente seguir o argumento de nome de nó a `@node` com até três argumentos opcionais no restante da mesma linha, separando os argumentos com vírgulas. Esses são os nomes dos ponteiros 'Próximo', 'Anterior', e 'Acima', nessa exata ordem. Recomenda-se omiti-los se o seu documento Texinfo for hierarquicamente organizado, como virtualmente todos são (veja-se Seção 4.8 [Criação de Ponteiros do `makeinfo`], Página 35).

Quaisquer espaços antes ou após cada nome na linha `@node` são ignorados.

O modelo para uma linha de nó completamente preenchida com os ponteiros 'Próximo', 'Anterior', e 'Acima' se parece com isto:

```
@node nome_do_nodo, next, previous, up
```

O argumento *nome_do_nodo* deve necessariamente estar presente, porém os outros são opcionais. Se você desejar especificar algum, porém não outros, apenas insira vírgulas conforme necessário, como em: '`@node meu_nodo,,,nodo_superior`'. Entretanto, recomenda-se deixar desativados todos os ponteiros e permitir que `makeinfo` os determine.

Se você estiver usando o GNU Emacs, você pode usar os comandos de atualização de nós disponibilizados pelo modo Texinfo para inserir os nomes dos ponteiros; ou (recomendado), você pode deixar os ponteiros fora do arquivo Texinfo e permitir que `makeinfo` insira os ponteiros de nó no arquivo Info que ele criar. (Veja-se Apêndice D [Modo Texinfo], Página 237, e Seção 4.8 [Criação de Ponteiros do `makeinfo`], Página 35).

Alternativamente, você pode inserir os ponteiros 'Próximo', 'Anterior', e 'Acima' você mesmo. Se você fizer isso, você pode achar de grande ajuda usar o comando de teclado do modo Texinfo `C-c C-c n`. Esse comando insere '`@node`' e uma linha de comentário que te ajuda a monitorar quais argumentos são para quais ponteiros. Essa linha de comentário é especialmente útil se você não estiver familiarizado com Texinfo.

4.4 Exigências de Linha de `@node`

Os nomes usados com `@node` tem várias exigências:

- Todos os nomes de nó em um arquivo Texinfo devem ser únicos.

  Isso significa, por exemplo, que, se você finalizar cada capítulo com um sumário, você precisa nomear cada nó de sumário diferentemente. Você não pode simplesmente chamá-los todos de "Sumário". Você pode, entretanto, duplicar os títulos dos capítulos, seções, e afins. Assim, você pode finalizar cada capítulo com uma seção chamada "Sumário", contanto que os nomes de nó para tais seções sejam todos diferentes.

- Os nomes de nó podem conter comandos @. A saída geralmente é o resultado natural do comando; por exemplo, usar `@TeX{}` em um nome de nó resulta no logotipo do TEX sendo exibido, conforme poderia estar em texto normal. As referências cruzadas deveriam usar `@TeX{}`, assim como o nome de nó usa.

  Para saída Info e HTML, especialmente, é necessário expandir os comandos para alguma sequência de caracteres planos; por exemplo, `@TeX{}` se expande para as três letras '`TeX`' no nome de nó do Info. Entretanto, as referências cruzadas ao nó não deveriam tomar o "atalho" de usar '`TeX`'; furar ao nome do nó real, comandos e tudo.

  Alguns comandos não fazem sentido em nomes de nó; por exemplo, ambientes (e.g., `@quotation`), comandos que leem uma linha inteira como seu argumento (e.g., `@sp`), e muitos outros.

Para a lista completa de comandos permitidos, e suas expansões para identificadores HTML e nomes de arquivo, veja-se Seção 22.4.3 [Expansão do Comando Xref do HTML], Página 199. As expansões para Info geralmente são dadas como a principal descrição do comando.

Antes do lançamento de Texinfo 5 em 2013, essa característica era suportada em uma maneira ad hoc (a opção `--commands-in-node-names` para `makeinfo`). Agora é parte da linguagem.

- Infelizmente, você não pode confiavelmente usar pontos, vírgulas ou dois pontos em um nome de nó; esses podem confundir o leitor Info. Também, um nome de nó não pode iniciar com um abre parênteses precedendo um fecha parênteses, como em `(não)permitido`, dado que tal sintaxe é usada para especificar um manual externo. (Talvez essas limitações sejam removidas algum dia).

  `makeinfo` alerta sobre tais usos problemáticos em nomes de nó, itens de menu e referências cruzadas. Se você não quiser ver os alertas, você pode configurar a variável de personalização `INFO_SPECIAL_CHARS_WARNING` para '`0`' (veja-se Seção 20.5.4 [Outras Variáveis de Personalização], Página 177).

  Também, se você insistir em usar tais caracteres em nomes de nó (aceitando a saída Info subpadrão resultante), com o objetivo de não confundir os processadores Texinfo, você deve ainda encapsular tais caracteres, usando ou inserções especiais (veja-se Seção 12.1.3 [Inserindo Uma Vírgula], Página 95) ou `@asis` (veja-se [`@asis`], Página 78). Por exemplo:

  ```
  @node foo@asis{::}bar
  ```

  Como um exemplo de se evitar os caracteres especiais, o seguinte é um título de seção neste manual:

  ```
  @section @code{@@unnumbered}, @code{@@appendix}: ...
  ```

  Porém, falta ao nome de nó correspondente as vírgulas e o subtítulo:

  ```
  @node @unnumbered @appendix
  ```

- Maiúsculas e minúsculas são significantes em nomes de nó.
- Os espaços antes e depois dos nomes na linha '`@node`' são ignorados. Caracteres espaços em branco múltiplos "dentro" de um nome são substituídos por um espaço único. Por exemplo:

  ```
  @node foo bar
  @node  foo bar,
  @node foo bar ,
  @node foo  bar,
  @node  foo  bar ,
  ```

  todos definem o mesmo nó, chamado '`foo bar`'. Em entradas de menu, esse é o nome que deveria ser usado: sem espaços antes ou depois, e um espaço interno único. (Para referências cruzadas, o nome de nó usado nos fontes Texinfo é automaticamente normalizado dessa maneira).
- Os ponteiros próximo/anterior/acima em linhas `@node` devem ser os nomes dos nós. (É recomendado deixar esses nomes de ponteiros de nó explícitos, os quais automaticamente evitam quaisquer problemas aqui; veja-se Seção 4.8 [Criação de Ponteiros do `makeinfo`], Página 35).

4.5 O Primeiro Nó

O primeiro nó de um arquivo Texinfo é o nó *Top*, exceto em um arquivo incluído (veja Capítulo 18 [Arquivos de Inclusão], Página 146). O nó Top deveria conter um sumário curto, permissões de cópia e um menu mestre. Veja-se Seção 3.6 [O Nó Top], Página 23, para mais informações sobre o conteúdo e exemplos do nó Top.

Aqui está uma descrição dos ponteiros de nó a serem usados no nó Top:

- O nó Top (o qual deve ser nomeado '`top`' ou '`Top`') deveria ter como seu nó 'Acima' o nome de um nó em outro arquivo, onde existe um menu que leva a esse arquivo. Especifique o nome de arquivo entre parênteses.

  Usualmente, todos os arquivos Info estão disponíveis por meio de uma árvore virtual única de Info, construída a partir de múltiplos diretórios. Nesse caso, use '`(dir)`' como o pai do nó Top; isso especifica o nó topo de nível no arquivo `dir`, o qual contém o menu principal para o sistema Info como um todo. (Cada diretório com arquivos Info deve conter um arquivo chamado `dir`).

  Isso está bom para Info, porém para saída HTML, alguém bem pode desejar que o link Acima oriundo do nó Top vá para algum lugar diferente de '`dir.html`'. Por exemplo, para GNU, o lugar natural seria `http://www.gnu.org/manual/` (uma página da web contendo links para a maior parte dos manuais GNU), melhor especificada que apenas `/manual/`, se o manual será instalado em `www.gnu.org`. Isso pode ser especificado com a variável de personalização `TOP_NODE_UP_URL` (veja-se Seção 20.5.3 [Variáveis de Personalização de HTML], Página 172), como em

  ```
  $ makeinfo --html -c TOP_NODE_UP_URL=/manual/ ...
  ```

  Todos os links para `(dir)` serão substituídos pela URL dada.
- O nó 'Anterior' do nó Top é usualmente ou omitido ou também configurado para `(dir)`. Qualquer uma está legal.
- O nó 'Próximo' do nó Top deveria ser o primeiro capítulo em seu documento.

Veja-se Seção 21.2 [Instalando Um Arquivo do Info], Página 188, para mais informação sobre a instalação de um arquivo Info no diretório `info`.

Geralmente, é melhor deixar os ponteiros completamente e deixar que as ferramentas os definam implicitamente, com este resultado simples:

```
@node Top
```

4.6 O Comando de Seccionamento `@top`

O comando `@top` é um comando especial de seccionamento que você deveria somente usar após uma linha '`@node Top`' no início de um arquivo Texinfo. O comando `@top` diz ao formatador `makeinfo` qual nó é para ser usado como raiz da árvore de nós.

O comando produz o mesmo tipo de saída que `@unnumbered` (veja-se Seção 5.4 [`@unnumbered @appendix`], Página 40).

O nó `@top` é convencionalmente envolto em um condicional `@ifnottex` de forma que não aparecerá na saída TEX (veja-se Capítulo 16 [Condicionais], Página 128). Assim, na prática, um nó Top geralmente se parece com isto:

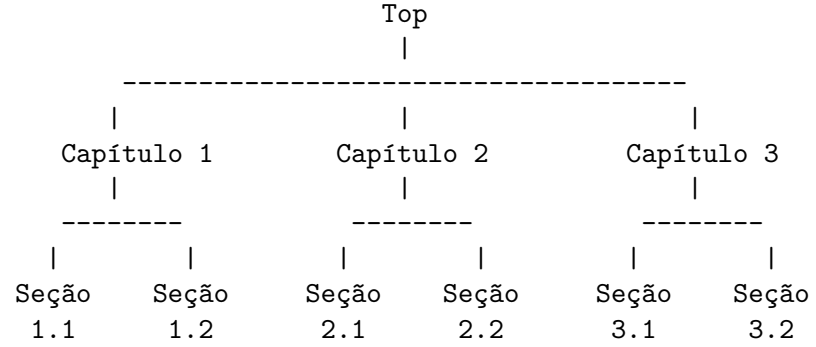
```
@ifnottex
@node Top
@top titulo-do-seu-manual

sumario-de-altissimo-nivel
@end ifnottex
```

`@top` é ignorado ao se levantar ou abaixar seções. Ou seja, nunca é baixado e nada pode ser levantado a ele (veja-se Seção 5.12 [Elevar/rebaixar seções], Página 44).

4.7 Ilustração de Menu e Nó

Aqui está uma diagrama que ilustra um arquivo Texinfo com três capítulos, cada um dos quais contendo duas seções.

A "raiz" está no topo do diagrama e as "folhas" estão na parte inferior. Assim é como tal diagrama é desenhado convencionalmente; ele ilustra uma árvore de cabeça para baixo. Por essa razão, o nó raiz é chamado de nó 'Top', e ponteiros de nó 'Acima' levam você para mais perto da raiz.



Usando ponteiros explícitos (não recomendado, mas mostrado para fins do exemplo), o comando totalmente escrito para iniciar o Capítulo 2 poderia ser isto:

```
@node     Capítulo 2,  Capítulo 3, Capítulo 1, Top
@comment  nome-nó,   next,       previous,   up
```

Essa linha `@node` diz que o nome desse nó é "Capítulo 2", o nome do nó 'Próximo' é "Capítulo 3", o nome do nó 'Anterior' é "Capítulo 1" e o nome do nó 'Acima' é "Top". Você pode (e deveria) omitir a explicitação desses nomes de nó se o seu documento estiver hierarquicamente organizado (veja-se Seção 4.8 [Criação de Ponteiros do `makeinfo`], Página 35), porém os relacionamentos de ponteiros ainda serão obtidas.

> **Nota:** 'Próximo' e 'Anterior' se referem a nós no *mesmo nível hierárquico* no manual, não necessariamente ao próximo nó dentro do arquivo Texinfo. No arquivo Texinfo, o nó subsequente pode estar a um nível mais baixo—um nó de nível de seção frequentemente segue um nó de nível de capítulo, por exemplo. (O nó 'Top' contém a exceção a essa regra. Dado que o nó 'Top' é o único nó naquele nível, 'Próximo' se refere ao primeiro nó seguinte, o qual quase sempre é um nó de capítulo ou nível de capítulo).

Para ir para as Seções 2.1 e 2.2 usando Info, você precisa de um menu dentro do Capítulo 2. (veja-se Seção 4.9 [Menus], Página 36). Você escreveria o menu pouco antes do início da Seção 2.1, como isto:

```
@menu
* Seção. 2.1::    Descrição dessa seção.
* Seção. 2.2::    Descrição.
@end menu
```

Usando ponteiros explícitos, o nó para a Seção. 2.1 é escrito como isto:

```
@node     Seção. 2.1, Seção. 2.2, Capítulo 2, Capítulo 2
@comment  nome-nó,  next,       previous,   up
```

Em formato Info, os ponteiros 'Próximo' e 'Anterior' de um nó geralmente conduzem à outros nós no mesmo nível—de capítulo para capítulo ou de seção para seção (as vezes, conforme mostrado, o ponteiro 'Anterior' aponta para cima); um ponteiro 'Acima' geralmente conduz à um nó ao nível acima (mais perto do nó 'Top'); e um 'Menu' conduz à nós a um nível abaixo (mais perto das 'folhas'). (Uma referência cruzada pode apontar para um nó em qualquer nível; veja-se Capítulo 6 [Referências Cruzadas], Página 45).

Um comando `@node` e um comando de estruturamento de capítulo são convencionalmente usados juntos, nessa ordem, frequentemente seguidos por comandos de indexação. (Conforme mostrado no exemplo acima, você pode seguir a linha `@node` com uma linha de comentário, por exemplo, para mostrar qual ponteiro é qual se ponteiros explícitos são usados). Os processadores Texinfo usam essa construção para determinar os relacionamentos entre nós e comandos de seccionamento.

Aqui está o início de um capítulo neste manual chamado "Finalizando um Arquivo Texinfo". Isso mostra uma linha `@node` seguida por uma linha `@chapter` e, então, por linhas de indexação.

```
@node Finalizando um Arquivo
@chapter Finalizando um Arquivo Texinfo
@cindex Finalizando um arquivo Texinfo
@cindex Finalização de arquivo Texinfo
@cindex Finalização de arquivo
```

Uma versão mais antiga do manual usava ponteiros de nó explícitos. Aqui está o início do mesmo capítulo para esse caso. Isto mostra uma linha `@node` seguida por uma linha de comentário, uma linha `@chapter` e, então, por linhas de indexação.

```
@node Finalizando um Arquivo, Estruturamento, Iniciando um Arquivo, Top
@comment nome-nó,          next,          previous,          up
@chapter Finalizando um Arquivo Texinfo
@cindex Finalizando um Arquivo Texinfo
...
```

4.8 Criação de Ponteiros do `makeinfo`

O programa `makeinfo` pode automaticamente determinar ponteiros de nó para um documento hierarquicamente organizado. Essa característica de criação implícita de ponteiro de nó em `makeinfo` liberta você da necessidade de atualizar menus e ponteiros manualmente ou com comandos no Modo Texinfo. (Veja Seção D.5 [Atualizando Nós e Menus], Página 240). Nós recomendamos enfaticamente tomar vantagem disso.

Para fazer isso, escreva suas linhas `@node` apenas com o nome do nó:

```
@node Meu Nó
```

Você não precisa escrever os ponteiros 'Próximo', 'Anterior', e 'Acima'.

Então, você deve escrever um comando de seccionamento, como `@chapter` ou `@section`, na linha imediatamente seguinte a cada linha `@node` truncada (exceto aquelas linhas de comentários que podem intervir). Isso é como normalmente ocorre.

Também, você deve escrever o nome de cada nó (exceto para o nó 'Top') em um menu que esteja um ou mais níveis hierárquicos acima do nível do nó.

Finalmente, você deve seguir a linha 'Top' da linha `@node` com uma linha iniciando com `@top` para marcar o nó de nível de topo no arquivo. Veja-se Seção 4.6 [Comando `@top`], Página 33.

Se você usa um menu detalhado no seu menu mestre (veja-se Seção 3.6.2 [Partes do Menu Mestre], Página 24), marque-o com o ambiente `@detailmenu ... @end detailmenu`, ou `makeinfo` ficará confuso, tipicamente acerca do último e/ou primeiro nó no documento.

Na maior parte dos casos, você desejará tomar vantagem dessa característica e não redundantemente especificar ponteiros de nó que os programas podem determinar. Entretanto, não se exige que os documentos Texinfo sejam organizados hierarquicamente ou, de fato, conterem quaisquer comandos de seccionamento (por exemplo, se você nunca pretender que o documento seja impresso), de forma que os ponteiros de nó ainda possam ser especificados explicitamente, em completa generalidade.

4.9 Menus

Menus contém ponteiros para subordinar nós. Em saída online, você usa menus para ir para tais nós. Os menus não tem efeito em manuais impressos e não aparecem neles.

4.9.1 Escrevendo um Menu

Um menu consiste de um comando `@menu` em uma linha própria, seguido por linhas de entrada de menu ou linhas de comentário de menu, e então seguido por um comando `@end menu` em uma linha própria.

Um menu se parece com isto:

```
@menu
Unidades de Texto Mais Largas

* Arquivos::                    Tudo sobre a manipulação de arquivos
* Múltiplos: Buffers.           Múltiplos buffers; editando
                                 vários arquivos de uma vez
@end menu
```

Em um menu, cada linha que inicie com um '`* `' é uma *entrada de menu*. (Note o espaço após o asterisco).

Uma linha que não se inicie com um '`* `' também pode aparecer em um menu. Tal linha não é uma entrada de menu, mas sim uma linha de *comentário de menu* que aparece no arquivo Info. No exemplo acima, a linha '`Unidades de Texto Mais Largas`' é uma linha de comentário de menu; as duas linhas iniciando com '`* `' são entradas de menu.

Tecnicamente, menus podem te transportar para qualquer nó, independentemente da estrutura do documento; mesmo para nós em um arquivo Info diferente. Entretanto, nós não recomendamos fazer uso disso, pois é trabalhoso para os leitores seguirem. Também, a característica de criação implícita de ponteiros do comando `makeinfo` (veja Seção 4.8 [Criação de Ponteiros do `makeinfo`], Página 35) e os comandos de atualização do modo Texinfo do GNU Emacs funcionam somente para criar menus de nós subordinados em um documentos hierarquicamente estruturado. É muito melhor usar referências cruzadas para se referir a nós arbitrários.

O comando `makeinfo` pode automaticamente gerar menus em nós para saída Info e HTML, baseado na estrutura de capítulo do documento. Para especificar que você deseja que o comando faça isso, coloque uma linha '`@validatemenus off`' próxima do início do documento.

Em Info, um usuário seleciona um nó o comando *`m`* (`Info-menu`). O nome de entrada de menu é o que o usuário digita após o comando *`m`*.

Na saída HTML oriunda de `makeinfo`, o atributo `accesskey` é usado com os valores '`1`'. . . '`9`' para as primeiras nove entradas. Isso permite que as pessoas usando navegadores da Internet sigam a primeira entrada de menu usando (tipicamente) *`M-digit`*, por exemplo, *`M-1`* para a primeira entrada.

4.9.2 Um Exemplo de Menu

Um menu se parece com isto em Texinfo:

```
@menu
* Nome de entrada do menu: Nome do nó. Uma descrição curta.
* Nome do nó::                Essa forma é preferida.
@end menu
```

Isso produz:

```
* menu:

* Nome de entrada do menu: Nome do nó. Uma descrição curta.
* Nome do nó::                Essa forma é preferida.
```

Aqui está um exemplo de como você poderá vê-lo em um arquivo Texinfo:

```
@menu
Unidades de Texto Mais Largas

* Arquivos::                    Tudo sobre a manipulação de arquivos.
* Múltiplos: Buffers.           Múltiplos buffers; editando
                                vários arquivos de uma vez.
@end menu
```

Isso produz:

```
* menu:
Unidades de Texto Mais Largas

* Arquivos::                    Tudo sobre a manipulação de arquivos.
* Múltiplos: Buffers.           Múltiplos buffers; editando
                                vários arquivos de uma vez.
```

Nesse exemplo, o menu tem duas entradas. '`Arquivos`' tanto é um nome de entrada de menu quanto o nome do nó referenciado por aquele nome. '`Múltiplos`' é o nome de entrada de menu; ele se refere ao nó chamado '`Buffers`'. A linha '`Unidades de Texto Mais Largas`' é um comentário; ele aparece no menu, porém não é uma entrada.

Dado que nenhum nome de arquivo é especificado com '`Arquivos`' ou '`Buffers`', eles podem ser os nomes de nós no mesmo arquivo Info (veja Seção 4.9.6 [Referenciando Outros Arquivos do Info], Página 38).

4.9.3 Local de Menu

Pode existir ao menos um menu em um nó. Um menu é convencionalmente localizado ao final de um nó, sem qualquer texto regular ou comandos adicionais entre o `@end menu` e o início do próximo nó.

Essa convenção é útil, dado que um leitor que usa o menu poderia facilmente se esquecer de quaisquer de tais textos. Também, quaisquer de tais textos pós menu serão considerados parte do menu na saída Info (a qual não tem marcador para o final de um menu). Assim, uma linha iniciando com '`* `' será igualmente incorretamente manipulada.

Geralmente é melhor se um nó com um menu não contenha muito texto. Se você se encontrar com um monte de texto antes de um menu, geralmente nós recomendamos mover tudo, menos um par de parágrafos, para um novo sub nó. Do contrário, é fácil para os leitores esquecerem o menu.

4.9.4 As Partes de um Menu

Uma entrada de menu tem três partes, das quais apenas a segunda é exigida:

1. O nome de entrada de menu (opcional).
2. O nome do nó (exigido).
3. Uma descrição do item (opcional).

O modelo para uma entrada de menu genérica se parece com isto (porém, veja a próxima seção para mais uma possibilidade):

```
* nome_de_entrada_de_menu: nome_de_nodo.   descrição
```

Coloque um único dois pontos após o nome de entrada de menu, seguido do nome de nó e um tab, vírgula, enter ou os dois caracteres "ponto" e "espaço" ('`. `').

A terceira parte de uma entrada de menu é uma frase descritiva ou sentença. Os nomes de entrada de menu e os nomes de nó frequentemente são curtos; a descrição explana para o(a)

leitor(a) a respeito de que se trata o nó. Uma descrição útil complementa o nome de nó em vez de repeti-lo. A descrição, a qual é opcional, pode ser espraiada por várias linhas; se isso acontecei, alguns autores preferem recuar a segunda linha enquanto outros preferem alinhá-lo com a primeira (e todas as outras). É completamente com você. Uma linha variz, ou a próxima entrada de menu, finaliza a descrição.

Os caracteres de espaço em um menu são preservados como estão na saída Info; isso permite formatar o menu como se desejar. Infelizmente, você deve digitar os nomes de nó sem quaisquer espaços extra ou algumas versões de alguns leitores Info não encontrarão o nó (veja Seção 4.4 [Exigências de Linha de Nó], Página 31).

`makeinfo` alerta quando o texto de um item de menu (e nomes de nó e referências cruzadas) contem uma construção problemática que interferirá com sua análise em Info. Se você não deseja ver esses alertas, você pode configurar a variável de personalização `INFO_SPECIAL_CHARS_WARNING` para '`0`' (veja Seção 20.5.4 [Outras Variáveis de Personalização], Página 177).

4.9.5 Entrada de Menu Menos Desordenada

Quando o nome de entrada de menu e o nome de nó são os mesmos, você pode escrever o nome imediatamente após o asterisco e espaço no início da linha e seguir com o nome com "dois pontos" duas vezes.

Por exemplo, escreva

```
* Nome::                       descrição
```

em vez de

```
* Nome: Nome.                  descrição
```

Nós recomendamos usar o nome de nó para o nome de entrada de menu sempre que possível, dado que isso reduz a confusão visual no menu.

4.9.6 Referenciando Outros Arquivos do Info

Você pode criar uma entrada de menu que habilite um leitor em Info a ir para um nó em outro arquivo Info, escrevendo o nome de arquivo entre parênteses um pouco antes do nome do nó. Alguns exemplos:

```
@menu
* nome-da-primeira-entrada:(nome-do-arquivo)nome-do-nó.     descrição
* (nome-do-arquivo)segundo-nó::                                  descrição
@end menu
```

Por exemplo, para se referir diretamente aos nós '`Outlining`' e '`Rebinding`' no *Manual do Emacs*, você poderia escrever um menu como este:

```
@menu
* Outlining: Modo Contorno(emacs). O principal modo de edição de
                                   contornos.

* (emacs)Rebinding::          Como redefinir o significado de uma tecla.

@end menu
```

Se você não listar o nome de nó, mas apenas nomear o arquivo, então Info presume que você está se referindo ao nó 'Top'. Exemplos:

```
* Info: (info).         Sistema de navegação de documentação.
* (emacs)::             Editor de texto, extensível, auto-documentador
```

Os comandos de atualização de menu do modo Texinfo do GNU Emacs somente funcionam com nós dentro do buffer atual, de forma que você não pode usá-los para criar menus que se refiram a outros arquivos. Você deve escrever tais menus manualmente.

5 Estruturamento de Capítulo

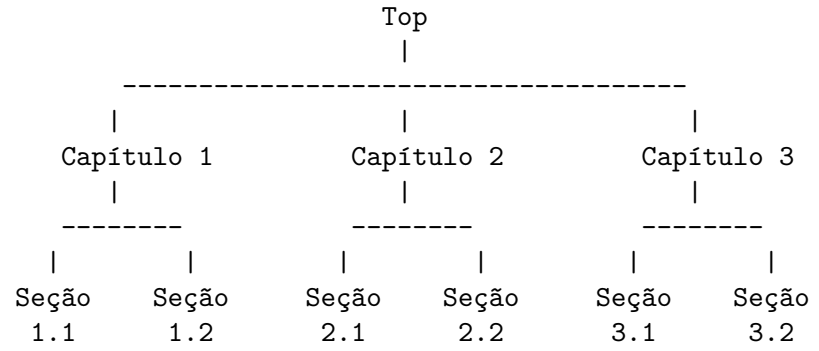
Os comandos de *Estruturamento de Capítulo* do Texinfo dividem um documento em uma hierarquia de capítulos, seções, subseções e subsubseções. Esses comandos geram cabeçalhos largos no texto, como o acima. Eles também proveem informação para a geração de índices (veja Seção 3.5 [Gerando Um Sumário], Página 22).

Normalmente você coloca um comando `@node` imediatamente antes de cada comando de Estruturamento de Capítulo. Veja Capítulo 4 [Nós], Página 29.

5.1 Estrutura de Árvore das Seções

Um arquivo Texinfo é geralmente estruturado como um livro com capítulos, seções, subseções, e afins. Essa estrutura pode ser visualizada como uma árvore (ou melhor, como uma árvore de cabeça para baixo) com a raiz no topo e as folhas correspondendo aos capítulos, seções, subseções e subsubseções.

Aqui está um diagrama que mostra um arquivo Texinfo com três capítulos, cada qual com duas seções.



Em um arquivo Texinfo que tenha essa estrutura, o início do Capítulo 2 poderia ser escrito assim:

```
@node    Capítulo 2
@chapter Capítulo 2
```

Para o propósito do exemplo, aqui está como poderia ser escrito com ponteiros de nó explícitos:

```
@node    Capítulo 2,  Capítulo 3, Capítulo 1, Top
@chapter Capítulo 2
```

Os comandos de estruturamento de capítulo estão descritos nas seções seguintes; o comando `@node` está descrito no capítulo anterior (veja Capítulo 4 [Nós], Página 29).

5.2 Tipos de Comandos Estruturantes

Os comandos de estruturamento de capítulo se dividem em quatro grupos ou séries, cada um dos quais contendo comandos de estruturamento correspondentes aos níveis hierárquicos de capítulos, seções, subseções e subsubseções.

Os quatro grupos de comandos são: a série `@chapter`; a série `@unnumbered`; a série `@appendix`; e a série `@heading`. Cada comando produz um título com uma aparência diferente no corpo do documento. Alguns dos comandos listam seus títulos no sumário, enquanto outros não. Aqui estão os detalhes:

- As séries de comandos `@chapter` e `@appendix` produzem entradas numeradas ou letradas, ambas no corpo de um documento e em seu sumário.

- A série de comandos `@unnumbered` produzem entradas não numeradas, ambas no corpo de um documento e em seu sumário. O comando `@top`, o qual tem um uso especial, é um membro dessa série (veja Seção 4.6 [Comando `@top`], Página 33). Uma seção `@unnumbered` é uma parte normal da estrutura do documento.
- A série de comandos `@heading` produzem cabeçalhos não numerados simples que não aparecem em um sumário, não são associados com nós e não podem ser referenciados. Esses comandos de cabeçalhos nunca iniciam uma nova página.

Quando um comando `@setchapternewpage` diz para fazer isso, ou seja, novo capítulo com nova página, os comandos `@chapter`, `@unnumbered` e `@appendix` iniciam novas páginas no manual impresso; os comandos `@heading` não. Veja Seção 3.7.2 [`@setchapternewpage`], Página 25.

Aqui está um sumário:

			Sem nova página
Numerada	*Não numerada*	*Letrada/numerada*	*Não numerada*
No conteúdo	No conteúdo	No conteúdo	Não no conteúdo
	`@top`		`@majorheading`
`@chapter`	`@unnumbered`	`@appendix`	`@chapheading`
`@section`	`@unnumberedsec`	`@appendixsec`	`@heading`
`@subsection`	`@unnumberedsubsec`	`@appendixsubsec`	`@subheading`
`@subsubsection`	`@unnumberedsubsubsec`	`@appendixsubsubsec`	`@subsubheading`

5.3 `@chapter`: Estruturamento de Capítulo

`@chapter` identifica um capítulo no documento–a nível mais alto da hierarquia de estruturamento normal do documento. Escreva o comando no início de uma linha e o siga na mesma linha pelo título ou capítulo. O capítulo é numerado automaticamente, iniciando a partir de 1.

Por exemplo, o capítulo presente neste manual é intitulado "`@chapter`: Estruturamento de Capítulo"; a linha `@chapter` se parece com isto:

```
@chapter @code{@@chapter}: Estruturamento de Capítulo
```

Em TEX, o comando `@chapter` produz um cabeçalho de capítulo no documento.

Em saída Info e texto plano, o comando `@chapter` faz com que o título apareça em uma linha própria, com uma linha de asteriscos inserida abaixo. Assim, o exemplo acima produz a seguinte saída:

```
5 Estruturamento de Capítulo
****************************
```

Em HTML, o comando `@chapter` produz um cabeçalho de nível `<h2>` por padrão (controlado pela variável de personalização `CHAPTER_HEADER_LEVEL`, veja Seção 20.5.4 [Outras Variáveis de Personalização], Página 177).

Na saída XML e Docbook, um elemento `<chapter>` é produzido, o qual inclui todas as seções seguintes, até o próximo capítulo.

5.4 `@unnumbered`, `@appendix`: Capítulos com Outra Rotulagem

Use o comando `@unnumbered` para iniciar um elemento de nível de capítulo que aparece sem números de capítulos de qualquer tipo. Use o comando `@appendix` para iniciar um apêndice que é rotulado por letras ('A', 'B', ...) em vez de números; apêndices também estão ao nível de estruturamento de capítulo.

Escreva um comando `@appendix` ou `@unnumbered` no início de uma linha e siga-o, na mesma linha, pelo título, exatamente como com `@chapter`.

Texinfo também provê um comando `@centerchap`, o qual é análogo a `@unnumbered`, porém centraliza seus argumentos nas saídas impressa e HTML. Esse tipo de escolha estilística geralmente não é oferecida por Texinfo. Ela pode ser adequada para documentos curtos.

Com `@unnumbered`, se o nome do nó associado é uma destas palavras do Inglês (não diferencia maiúsculas de minúsculas):

```
Acknowledgements  Colophon  Dedication  Preface
```

então a saída Docbook usa etiquetas especiais correspondentes (`<preface>`, etc.) em vez do padrão `<chapter>`. O argumento ao próprio `@unnumbered` pode ser qualquer coisa, e é produzido como o texto `<title>` a seguir como de costume.

5.5 `@majorheading`, `@chapheading`: Cabeçalhos de Nível de Capítulo

Os comandos `@majorheading` e `@chapheading` produzem cabeçalhos semelhantes a capítulos no corpo de um documento.

Entretanto, nenhum desses comandos produz uma entrada no índice, e nenhum desses comandos faz com que TEX inicie uma nova página em um manual impresso.

Em TEX, um comando `@majorheading` gera um espaço em branco vertical mais largo, antes do cabeçalho, que um comando `@chapheading`, porém, de outra forma, é o mesmo.

Em Info e texto plano, os comandos `@majorheading` e `@chapheading` produzem a mesma saída que `@chapter`: o título é impresso em uma linha própria com uma linha de asteriscos abaixo. Semelhante para HTML. A única diferença é a falta de numeração e a falta de qualquer associação com Nós. Veja Seção 5.3 [`@chapter`], Página 40.

5.6 `@section`: Seções Abaixo de Capítulos

Um comando `@section` identifica uma seção dentro de uma unidade de capítulo, seja ela criada com `@chapter`, `@unnumbered`, ou `@appendix`, seguindo o esquema de numeração do comando de nível de capítulo. Assim, dentro de um Capítulo `@chapter` numerado '1', as seções são numeradas '1.1', '1.2', etc.; dentro de um "capítulo" `@appendix` rotulado 'A', as seções são numeradas 'A.1', 'A.2', etc.; dentro de um capítulo `@unnumbered`, a seção não é numerada. A saída é sublinhada com '`=`' em Info e texto plano.

Para criar uma seção, escreva o comando `@section` no início de uma linha e siga-o, na mesma linha, pelo título da seção. Por exemplo,

```
@section Isto é uma seção
```

pode produzir o seguinte em Info:

```
5.7 Isto é uma seção
====================
```

Os títulos de seção são listados no índice.

A saída TEX, HTML, Docbook e XML é toda análoga à saída de nível de capítulo, apenas "um nível abaixo"; veja Seção 5.3 [`@chapter`], Página 40.

5.7 `@unnumberedsec`, `@appendixsec`, `@heading`

Os comandos `@unnumberedsec`, `@appendixsec` e `@heading` são respectivamente, os equivalentes não numerado, apêndice e cabeçalho do comando `@section` (veja a seção anterior).

`@unnumberedsec` e `@appendixsec` não precisam ser usados em circunstâncias normais, pois `@section` também pode ser usado dentro de capítulos `@unnumbered` e `@appendix`; novamente, veja a seção anterior.

`@unnumberedsec`

O comando `@unnumberedsec` pode ser usado dentro de um capítulo não numerado ou dentro de um capítulo ou apêndice normal para produzir uma seção não numerada.

`@appendixsec`
`@appendixsection`
: `@appendixsection` é a escrita mais longa do comando `@appendixsec`; os dois são sinônimos.

  Convencionalmente, o comando `@appendixsec` ou `@appendixsection` é usado somente dentro de apêndices.

`@heading` Você pode usar o comando `@heading` (quase) em qualquer lugar para um cabeçalho estilo seção que não aparecerá no índice. Os comandos da série `@heading` podem aparecer dentro da maior parte dos ambientes, por exemplo, ao longo de localizações patológicas e inúteis como dentro de `@titlepage`, como um argumento para outro comando, etc., não são permitidos.

5.8 `@subsection`: Subseções Abaixo de Seções

Subseções estão para seções como seções estão para capítulos; veja Seção 5.6 [`@section`], Página 41. Em Info e texto plano, os títulos de subseções são sublinhados com '`-`'. Por exemplo,

```
@subsection Esta é uma subseção
```

pode produzir

```
1.2.3 Esta é uma subseção
-------------------------
```

Os títulos das subseções são listados no índice.

A saída TEX, HTML, Docbook e XML é toda análoga à saída de nível de capítulo, apenas "dois níveis abaixo"; veja Seção 5.3 [`@chapter`], Página 40.

5.9 Os Comandos do Tipo `@subsection`

Os comandos `@unnumberedsubsec`, `@appendixsubsec` e `@subheading` são, respectivamente, os equivalentes não numerado, tipo apêndice e tipo cabeçalho do comando `@subsection`. (Veja Seção 5.8 [`@subsection`], Página 42).

`@unnumberedsubsec` e `@appendixsubsec` não precisam ser usados em circunstâncias normais, pois `@subsection` também pode ser usado dentro de seções de capítulos `@unnumbered` e `@appendix` (veja Seção 5.6 [`@section`], Página 41).

Um comando `@subheading` produz um cabeçalho semelhante àquele de uma subseção, exceto que não é numerado e não aparece no índice. Similarmente, um comando `@unnumberedsubsec` produz um cabeçalho não numerado semelhante àquele de uma subseção e um comando `@appendixsubsec` produz um cabeçalho do tipo subseção rotulado com uma letra e números; ambos esses comandos produzem cabeçalhos que aparecem no índice. Em Info e texto plano, os comandos do tipo `@subsection` geram um título sublinhado com hifens.

5.10 `@subsection` e Outros Comandos Subsub

O quarto e menor nível de comandos de seccionamento em Texinfo são os comandos 'subsub'. São eles:

`@subsubsection`
: Subsubseções estão para subseções como subseções estão para seções. (Veja Seção 5.8 [`@subsection`], Página 42). Os títulos das subsubseções aparecem no índice.

`@unnumberedsubsubsec`
: Os títulos de subsubseções não numeradas aparecem no índice, porém faltam números. Do contrário, subsubseções não numeradas são o mesmo que subsubseções.

`@appendixsubsubsec`
: Convencionalmente, os comandos de apêndice são usados somente para apêndices e são indicados e numerados apropriadamente. Eles também aparecem no índice.

`@subsubheading`
: O comando `@subsubheading` pode ser usado em qualquer lugar onde você desejar um pequeno cabeçalho que não aparecerá no índice.

Como com subseções, `@unnumberedsubsubsec` e `@appendixsubsubsec` não precisam ser usados em circunstâncias normais, pois `@subsubsection` também pode ser usado dentro de subseções de capítulos `@unnumbered` e `@appendix` (veja Seção 5.6 [`@section`], Página 41).

Em Info, os títulos 'subsub' são sublinhados com pontos. Por exemplo,

```
@subsubsection Esta é uma subsubseção
```

pode produzir

```
1.2.3.4 Esta é uma subsubseção
.............................
```

A saída TEX, HTML, Docbook e XML é toda análoga à saída de nível de capítulo, apenas "três níveis abaixo"; veja Seção 5.3 [`@chapter`], Página 40.

5.11 `@part`: Grupos de Capítulos

O comando de seccionamento final é o `@part`, para marcar uma *parte* de um manual, isto é, um grupo de capítulos ou (raramente) apêndices. Isso se comporta bem diferentemente de outros comandos de seccionamento, para se encaixar com a maneira que tais "partes" são convencionalmente usadas em livros.

Nenhum comando `@node` é associado com `@part`. Apenas escreva o comando em uma linha própria, incluindo o título da parte, no lugar no documento que você desejar marcar como o início daquela parte. Por exemplo:

```
@part Parte I:@* O Início
```

Como pode ser inferido desse exemplo, nenhuma numeração ou rotulamento automático do texto do `@part` é feito. O texto é tomado como está.

Por causa de as partes não serem associadas com nós, nenhum texto geral pode seguir a linha `@part`. Para produzir a saída pretendida, a linha deve ser seguida por um comando de nível de capítulo (incluindo seu nó). Assim, para continuar o exemplo:

```
@part Parte I:@* O Início

@node Introdução
@chapter Introdução
...
```

Na saída TEX, o texto `@part` é incluído em ambos os índices, o normal e o resumido, (veja Seção 3.5 [Conteúdo], Página 22), sem um número de página (dado que é a convenção normal). Adicionalmente, uma "página da parte" é saída no corpo do documento, apenas contendo o texto `@part`. No exemplo acima, o `@*` causa uma quebra de linha no página da parte (porém, é substituída com um espaço no índice). Essa página da parte é sempre forçada a estar em uma página ímpar (a direita), independentemente da capitulação do capítulo (veja Seção 3.7.2 [`@setchapternewpage`], Página 25).

Na saída HTML, o texto `@part` é similarmente incluído no índice, e um cabeçalho é incluído no texto do documento principal, como parte do nó do capítulo ou apêndice seguinte.

Na saída XML e Docbook, o elemento `<part>` inclui todos os capítulos seguintes, até o próximo `<part>`. Um `<part>` contendo capítulos também é fechado em um apêndice.

Na saída Info e texto plano, `@part` não tem efeito.

`@part` é ignorado quando do levantamento ou do abaixamento de seções (veja a próxima seção). Isto é, nunca é abaixado e nada pode ser levantado para ele.

5.12 Elevar/rebaixar seções: `@raisesections` e `@lowersections`

Os comandos `@raisesections` e `@lowersections` implicitamente levantam e abaixam o nível hierárquico dos seguintes capítulos, seções e dos outros comandos de seccionamento (excluindo as partes).

Isto é, o comando `@raisesections` modifica seções para capítulos, subseções para seções, e assim por diante. Por outro lado, o comando `@lowersections` modifica capítulos para seções, seções para subseções, e assim por diante. Assim, um comando `@lowersections` cancela um comando `@raisesections`, e vice versa.

Você pode usar `@lowersections` para incluir texto escrito como um arquivo Texinfo externo ou autônomo em outro arquivo Texinfo como um arquivo interno, arquivo included (veja Capítulo 18 [Arquivos de Inclusão], Página 146). O uso típico se parece com isto:

```
@lowersections
@include algum_arquivo.texi
@raisesections
```

(Sem o `@raisesections`, todas as seções subsequentes no arquivo principal também poderiam ser rebaixadas).

Se o arquivo included sendo rebaixado tiver um nó `@top`, você precisará condicionar sua inclusão com um parâmetro (veja Seção 16.5.1 [`@set @value`], Página 132).

De um ponto de vista prático, você geralmente só quer elevar ou rebaixar grandes blocos, geralmente em arquivos externos, como mostrado acima. O resultado final tem que ter menus que levem em conta a elevação e o rebaixamento, de forma que você não pode simplesmente intercalar arbitrariamente os comandos `@raisesections` e `@lowersections` por todo o documento.

O uso repetido dos comandos continua a elevar ou a rebaixar o nível hierárquico um passo de cada vez. Uma tentativa de elevar 'capítulo' acima reproduz comandos de capítulo; uma tentativa de rebaixar 'subsubseção' abaixo reproduz comandos subsubseção. Também, subsubseções rebaixadas e capítulos elevados não funcionarão com a característica de `makeinfo` de implicitamente determinar ponteiros de nó, dado que a estrutura de menu não pode ser representada corretamente.

Escreva cada comando `@raisesections` e `@lowersections` em uma linha própria.

6 Referências Cruzadas

Referências cruzadas são usadas para guiar o(a) leitor(a) para outras partes do mesmo ou de diferentes arquivos Texinfo.

6.1 Para Que São As Referências Cruzadas

Frequentemente, porém nem sempre, um documento impresso deveria ser projetado de forma que possa ser lido sequencialmente. As pessoas se cansam de ir e vir para encontrar informações que deveriam ser apresentadas a elas conforme necessário.

Entretanto, em qualquer documento, algumas informações estarão detalhadas demais para o contexto atual, ou incidental; use referências cruzadas para fornecer acesso a tais informações. Também, um sistema de ajuda online ou um manual de referência não é como um romance; poucos leem tais documentos em sequência do início ao final. Em vez disso, as pessoas buscam o que precisam. Por essa razão, tais criações deveriam conter muitas referências cruzadas para ajudar os leitores a achar outra informação que eles podem não ter lido.

Em um manual impresso, uma referência cruzada resulta em uma referência de página, a menos que seja para outro manual, caso no qual a referência cruzada nomeia esse manual. Em Info, uma referência cruzada resulta em uma entrada que você pode seguir usando o comando '`f`' (Veja Seção "Following cross-references" em *Info*.). Em HTML, uma referência cruzada resulta em um hiperlink.

Os vários comandos de referências cruzadas usam nós (ou âncoras, veja Seção 6.8 [`@anchor`], Página 51) para definir localizações de referências cruzadas. TEX precisa de nós para definir localizações de referências cruzadas. Quando TEX gera um arquivo DVI, grava cada número de página do nó e usa os números de página na produção das referências. Assim, mesmo que você esteja escrevendo um manual que será somente impresso, e não usado online, você deve, ainda assim, escrever linhas `@node` para nomear os lugares para os quais você faz referências cruzadas.

6.2 Diferentes Comandos de Referência Cruzada

Existem três diferentes comandos de referência cruzada:

`@xref` Usado para iniciar uma frase no manual impresso e em HTML com 'Veja ...' ou uma referência cruzada Info dizendo '`*Nota` *`name`*`: `*`node`*`.`'.

`@ref` Usado dentro de, ou mais frequentemente, ao final de uma frase; produz apenas a referência no manual impresso e em HTML, sem ser precedida por 'Veja' (mesmo que `@xref` para Info).

`@pxref` Usado dentro de parênteses, ao final de uma frase, ou, do contrário, antes de pontuação, para fazer uma referência. Sua saída inicia com um 'veja' escrito com letras todas minúsculas no manual impresso e em HTML, e um '`*nota`' também escrito com letras todas minúsculas em Info. ('`p`' é para 'parênteses').

Adicionalmente, existem comandos que produzem referências a documentos fora do sistema Texinfo. O comando `@cite` é usado para fazer referências a livros e manuais. `@url` produz um URL, por exemplo, uma referência a uma página na Rede Mundial de Computadores. `@inforef` é usado para fazer uma referência a um arquivo Info para o qual não existe manual impresso.

6.3 Partes de Uma Referência Cruzada

Um comando de referência cruzada exige apenas um argumento, o qual é o nome do nó para o qual se refere. Aqui está um exemplo simples:

```
@xref{Nome do nó}.
```

Em saída Info, isso produz

```
*Nota Nome do nó::.
```

Em um manual impresso, a saída é

> Veja Seção *nnn* [Nome do nó], página *ppp*.

Um comando de referência cruzada pode conter até quatro argumentos adicionais. Usando esses argumentos, você pode fornecer um nome de referência cruzada, uma descrição de tópico ou título de seção para a saída impressa, o nome de um arquivo diferente de manual e o nome de um manual diferente impresso. Para se referir a outro manual como um todo, o arquivo de manual e/ou o nome do manual impresso são os únicos argumentos exigidos (veja Seção 6.5 [Referenciando Um Manual Como Um Todo], Página 49).

Aqui está um exemplo de uma referência cruzada completa de cinco partes:

```
@xref{Nome do nó, Rótulo Online, Rótulo Impresso,
nome-arquivo-info, Um Manual Impresso}, para detalhes.
```

o qual produz

```
*Nota Rótulo Online: (nome-arquivo-info)Nome do nó,
para detalhes.
```

em Info e

> Veja seção "Rótulo Impresso" em *Um Manual Impresso*, para detalhes.

em um livro impresso.

Os cinco argumentos possíveis para uma referência cruzada são:

1. O nome do nó ou âncora (exigido, exceto para referência a manuais inteiros). Isso é a localização para onde a referência cruzada leva você. Em um documento impresso, a localização do nó fornece a referência de página apenas para referências dentro do mesmo documento. Use `@node` para definir o nó (veja Seção 4.3 [Escrevendo um Nó], Página 30), ou `@anchor` (veja Seção 6.8 [`@anchor`], Página 51).

   Escreva um nome de nó em uma referência cruzada exatamente da mesma maneira que na linha `@node`, incluindo a mesma capitalização; do contrário, os formatadores podem não encontrar a referência.
2. Um rótulo para saída online. Ele geralmente é omitido; então a descrição do tópico (terceiro argumento) é usado foi especificado; se também foi omitido, o nome do nó é usado.
3. Um rótulo para saída impressa. Frequentemente, isso é o título ou tópico da seção. Isso é usado como o nome da referência no manual impresso. Se omitido, o nome do nó é usado.
4. O nome do arquivo de manual no qual a referência está localizada, se for diferente o arquivo atual. Esse nome é usado por Info e por HTML.
5. O nome de um manual impresso oriundo de um diferente arquivo Texinfo.

O modelo para uma referência cruzada de cinco argumentos completa é semelhante a este

```
@xref{nome-do-nó, rótulo-online, rótulo-impresso,
nome-arquivo-info, título-manual-impresso}
```

Espaços em branco antes ou depois das vírgulas separando esses argumentos são ignorados. Para incluir uma vírgula em um dos argumentos, use `@comma{}` (veja Seção 12.1.3 [Inserindo Uma Vírgula], Página 95).

Quando do processamento com TeX, uma vírgula é automaticamente inserida após o número da página para referências cruzadas para dentro do mesmo manual, a menos que a chave de fechamento do argumento seja seguida por espaços não brancos (como uma vírgula ou um ponto). Isso lhe dá opção de ter uma vírgula lá na saída Info ou HTML. Por exemplo,

```
@xref{Outra Seção} para mais informação
```

produz 'Veja Outra Seção, página *ppp*, para mais informação' na saída impressa, e '`*Nota Outra Seção:: para mais informação`' na saída Info.

Se uma vírgula indesejada for adicionada, siga o argumento com um comando como '`@:`'. Por exemplo, '`@xref{Furacões}@: --- para os detalhes`' produz

> Veja Furacões, página *ppp* — para os detalhes

em vez de 'Veja Furacões, página *ppp*, — para os detalhes'.

Referências cruzados com um, dois, três quatro e cinco argumentos estão descritas separadamente seguindo a descrição de `@xref`.

`makeinfo` alerta quando o texto de uma referência cruzada (e nomes de nó e itens de menu) contém uma construção problemática que interferirá com sua análise em Info. Se você não deseja ver tais alertas, você pode configurar a variável de personalização `INFO_SPECIAL_CHARS_WARNING` para '`0`' (veja Seção 20.5.4 [Outras Variáveis de Personalização], Página 177).

6.4 `@xref`

O comando `@xref` gera uma referência cruzada para o início de uma frase.

6.4.1 `@xref` com Um Argumento

A forma mais simples do `@xref` pega um argumento, o nome de outro nó no mesmo arquivo Texinfo.

Por exemplo,

```
@xref{Tempestades Tropicais}.
```

produz

```
*Nota Tempestades Tropicais::.
```

em Info e

> Veja Seção 3.1 [Tempestades Tropicais], página 24.

em um manual impresso.

6.4.2 `@xref` com Dois Argumentos

Com dois argumentos, o segundo é usado como um rótulo para a saída online.

O modelo é semelhante a isto:

```
@xref{nome-nó, rótulo-online}.
```

Por exemplo,

```
@xref{Efeitos Elétricos, Relâmpago}.
```

produz:

```
*Nota Relâmpago: Efeitos Elétricos.
```

em Info e

> Veja Seção 5.2 [Efeitos Elétricos], página 57.

em um manual impresso, onde o nome do nó é impresso.

O segundo argumento para referências cruzadas deve observar algumas das restrições para nomes de nó (veja Seção 4.4 [Exigências de Linha de Nó], Página 31). O problema mais comum é que dois pontos não podem ser usados, pois isso interfere na análise do arquivo Info.

6.4.3 `@xref` com Três Argumentos

Um terceiro argumento substitui o nome de nó na saída TEX. O terceiro argumento deveria ser o nome da seção na saída impressa, ou, pelo contrário, indicar o tópico debatido pela aquela seção.

O modelo é semelhante a isto:

```
@xref{nome-nó, rótulo-online, rótulo-impresso}.
```

Por exemplo,

```
@xref{Efeitos Elétricos, Relâmpago, Trovão e Relâmpago}, para
detalhes.
```

produz:

```
*Nota Relâmpago: Efeitos Elétricos, para detalhes.
```

em Info e

> Veja Seção 5.2 [Trovão e Relâmpago], página 57, para detalhes.

em um manual impresso.

Se um terceiro argumento for dado e o segundo estiver vazio, então e terceiro argumento serve a ambos. (Note como duas vírgulas, lado a lado, marcam o segundo argumento vazio).

```
@xref{Efeitos Elétricos, , Trovão e Relâmpago}, para detalhes.
```

produz:

```
*Nota Trovão e Relâmpago: Efeitos Elétricos, para detalhes.
```

em Info e

> Veja Seção 5.2 [Trovão e Relâmpago], página 57, para detalhes.

em um manual impresso.

O terceiro argumento para referências cruzadas deve observar algumas das restrições para nomes de nó (veja Seção 4.4 [Exigências de Linha de Nó], Página 31). O problema mais comum é que dois pontos não podem ser usados, pois isso interfere na análise do arquivo Info.

De um ponto de vista prático, geralmente é melhor escrever referências cruzadas apenas com o primeiro argumento se o nome do nó e o título da seção forem iguais (ou quase), e com o primeiro e terceiro argumentos somente se o nome de nó e título forem diferentes.

Texinfo oferece uma configuração para usar o título da seção em vez dos nomes de nó por padrão em referências cruzadas (um terceiro argumento explicitamente especificado ainda tem precedência):

```
@xrefautomaticsectiontitle on
```

Tipicamente essa linha deveria ser dada próxima do início do documento e usada para o manual inteiro. Porém, você pode desativá-la se você desejar (`@xrefautomaticsectiontitle off`), por exemplo, se você estiver incluindo algum outro subdocumento que não tenha nomes de seção adequados.

6.4.4 `@xref` com Quatro e Cinco Argumentos

Em uma referência cruzada, um quarto argumento especifica o nome de outro arquivo Info, diferente do arquivo no qual a referência aparece, e um quinto argumento especifica seu título como um manual impresso.

O modelo completo é:

```
@xref{nome-nó, rótulo-online, rótulo-impresso,
nome-arquivo-info, título-manual-impresso}.
```

Por exemplo,

```
@xref{Efeitos Elétricos, Relâmpago, Trovão e Relâmpago, clima, Uma
```

```
Introdução à Meteorologia}.
```

produz esta saída em Info:

```
*Nota Relâmpago: (clima)Efeitos Elétricos.
```

Como você pode ver, o nome do arquivo Info está entre parênteses e precede o nome do nó.

Em um manual impresso, a referência se parece com isto:

> Veja seção "Trovão e Relâmpago" em *Uma Introdução à Meteorologia*.

O título do manual impresso é digitado como `@cite`; e a referência não possui um número de página, pois TEX não tem como saber para qual página uma referência se refere quando essa referência é para outro manual.

Próximo caso: frequentemente você deixará de fora o segundo argumento quando usar a versão longa de `@xref`. Nesse caso, o terceiro argumento, a descrição do tópico, será usada como o nome da referência cruzada em Info. Por exemplo,

```
@xref{Efeitos Elétricos, , Trovão e Relâmpago, clima, Uma Introdução à
Meteorologia}.
```

produz:

```
*Nota Trovão e Relâmpago: (clima)Efeitos Elétricos.
```

em Info e

> Veja seção "Trovão e Relâmpago" in *Uma Introdução à Meteorologia*.

em um manual impresso.

Próximo caso: se o nome do nó e o título de seção forem iguais no outro manual, você também pode deixar de fora o título da seção. Nesse caso, o nome de nó é usado em ambas as instâncias. Por exemplo,

```
@xref{Efeitos Elétricos,,, clima, Uma Introdução à Meteorologia}.
```

produz:

```
*Nota (clima)Efeitos Elétricos::.
```

em Info e

> Veja seção "Efeitos Elétricos" em *Uma Introdução à Meteorologia*.

em um manual impresso.

Um caso muito incomum: você pode querer se referir a outro arquivo de manual que esteja dentro de um manual impresso único—quando múltiplos arquivos Texinfo são incorporados à mesma execução de TEX, porém pode criar arquivos de saída Info ou HTML separados. Nesse caso, você precisa especificar apenas o quarto argumento e não o quinto.

Finalmente, também é permitido deixar de fora todos os argumentos, *exceto* o quarto e quinto, para referenciar outro manual como um todo. Veja a próxima seção.

6.5 Referenciando Um Manual Como Um Todo

Normalmente, você sempre deve nomear um nó em uma referência cruzada. Entretanto, não é incomum querer referenciar outro manual como um todo, em vez de uma seção em particular dentro dele. Nesse caso, dar qualquer nome de seção é uma distração desnecessária.

Assim, com referências cruzadas a outros manuais (veja Seção 6.4.4 [Quatro e Cinco Argumentos], Página 48), se o primeiro argumento for ou '`Top`' (escrito com a primeira letra maiúscula) ou inteiramente omitido, e o terceiro argumento for omitido, a saída impressa não inclui nome de nó ou seção. (A saída Info inclui '`Top`' se foi dado). Por exemplo,

```
@xref{Top,,, make, O Manual do GNU Make}.
```

produz:

```
*Nota (make)Top::.
```

e

> Veja *O Manual do GNU Make*.

Os leitores Info irão para o nó Top do manual se o nó 'Top' for explicitamente especificado ou não.

Também é possível (e é prática histórica) referenciar uma manual inteiro especificando-se o nó 'Top' e uma entrada apropriada para o terceiro argumento ao comando `@xref`. Usando esse idioma, para fazer uma referência cruzada para o *O Manual do GNU Make*, você poderia escrever:

```
@xref{Top,, Visão Geral, make, O Manual do GNU Make}.
```

o qual produz:

```
*Nota Visão Geral: (make)Top.
```

em Info e

> Veja seção "Visão Geral" in *O Manual do GNU Make*.

em um manual impresso.

Nesse exemplo, '`Top`' é o nome do primeiro nó e '`Visão Geral`' é o nome da primeira seção do manual. Não existe convenção amplamente usada para nomear a primeira seção em um manual impresso; isso é apenas o que o manual do Make usa. Essa arbitrariedade do primeiro nome é a razão principal pela qual é preferível omitir o terceiro argumento em referências cruzadas de manual inteiro.

6.6 `@ref`

`@ref` é aproximadamente o mesmo que `@xref`, exceto que não gera um 'Veja' na saída impressa, apenas a própria referência. Isso a torna útil como a última parte de uma frase.

Por exemplo,

```
Para mais informação, @pxref{Isso}, e @ref{Aquilo}.
```

produz em Info:

```
Para mais informação, *nota Isso::, e *nota Aquilo::.
```

e em saída impressa:

> Para mais informação, veja Seção 1.1 [Isso], página 1, e Seção 1.2 [Aquilo], página 2.

O comando `@ref` pode instigar os escritores a se expressarem de uma maneira que seja adequada para um manual impresso, porém parecerem estranhos no formato Info. Tenha em mente que seu público pode estar usando tanto o formato impresso quanto o Info. Por exemplo:

```
Os surtos de mar estão descritos na @ref{Furacões}.
```

aparenta ok na saída impressa:

> Os surtos de mar estão descritos na Seção 6.7 [Furacões], página 72.

porém, é estranho ler em Info, "nota" sendo um verbo:

```
Os surtos de mar estão descritos na *nota Furacões::.
```

6.7 `@pxref`

O comando de referência entre parênteses, `@pxref`, é aproximadamente o mesmo que `@xref`, porém é melhor usado no final de uma frase ou antes de um parêntese de fechamento. O comando difere do `@xref` em que TEX digita a referência para o manual impresso com um 'veja' escrito em letras minúsculas em vez de 'Veja' escrito com a primeira letra maiúscula.

Com um argumento, uma referência cruzada entre parênteses se parece com isto:

```
... tempestades causam inundações (@pxref{Furacões}) ...
```

o qual produz:

```
... tempestades causam inundações (*nota Furacões::) ...
```

em Info e

... tempestades causam inundações (veja Seção 6.7 [Furacões], página 72) ...

em um manual impresso.

Sem argumentos, uma referência cruzada entre parênteses tem este modelo:

```
... (@pxref{nome-nó, nome-referência-cruzada})
...
```

o qual produz

```
... (*nota nome-referência-cruzada: nome-nó.) ...
```

em Info e

... (veja Seção *nnn* [*nome-nó*], página *ppp*) ...

em um manual impresso.

`@pxref` pode ser usado com até cinco argumentos, exatamente como `@xref` (veja Seção 6.4 [`@xref`], Página 47).

Em versões antigas de Texinfo, não era permitido escrever pontuação após um `@pxref`, de forma que o comando poderia ser usado *somente* antes de um parêntese de fechamento. Esse não mais é o caso, de forma que agora o comando pode ser usado (por exemplo) no final de uma frase, onde um "'veja' escrito com letras minúsculas funciona melhor. Por exemplo:

```
... Para mais informação, @pxref{Mais}.
```

o qual tem a seguinte saída (em Info):

```
... Para mais informação, *nota Mais::.
```

Por uma questão de estilo, `@pxref` é mais usado nos finais de frase. Embora tecnicamente o comando funcione no meio de uma frase, essa localização interrompe o fluxo de leitura.

6.8 `@anchor`: Definindo Alvos Arbitrários de Referências Cruzadas

Uma *âncora* é uma posição no teu documento, rotulada de forma que referências cruzadas possam referenciá-la, exatamente como podem a nós. Você cria uma âncora com o comando `@anchor`, e dá o rótulo como um argumento normal delimitado por chaves. Por exemplo:

```
Isso marca o ponto @anchor{x-spot}.
...
@xref{x-spot,,o ponto}.
```

produz:

```
Isso marca o ponto.
...
Veja [o ponto], página 1.
```

Como você pode ver, o próprio comando `@anchor` não produz saída. Esse exemplo define uma âncora 'x-spot' exatamente antes da palavra 'ponto'. Você referenciá-la depois com um `@xref` ou outro comando de referência cruzada, conforme mostrado (veja Capítulo 6 [Referências Cruzadas], Página 45).

É melhor colocar os comandos `@anchor` exatamente antes da posição de que deseja referenciar; dessa maneira o olho do(a) leitor(a) é guiado ao texto correto quando pular para a âncora. Você pode colocar o comando `@anchor` em uma linha própria se isso ajudar a legibilidade do fonte. Espaços em branco (incluindo novas linhas) são ignorados após `@anchor`.

Os nomes de âncora e os nomes de nós não podem conflitar. Às âncoras e aos nós são dados tratamento similar de algumas maneiras; por exemplo, o comando `goto-node` recebe ou um nome de âncora ou um nome de nó como argumento. (Veja Seção "Ir para o nó" em *Info*).

Também, como nomes de nó, os nomes de âncora não podem incluir alguns caracteres (veja Seção 4.4 [Exigências de Linha de Nó], Página 31).

Por causa dessa dualidade, quando você deleta ou renomeia um nó, geralmente é uma boa ideia definir um `@anchor` com o nome antigo. Dessa maneira, quaisquer vínculos ao nó antigo, seja originários de outros manuais Texinfo ou páginas web em geral, continuam funcionando. Você também pode fazer isso com a característica `RENAMED_NODES_FILE` do `makeinfo` (veja Seção 22.4.7 [Preservação de Link Xref do HTML], Página 203). Ambos os métodos mantém links na web funcionando; a única diferença substantiva é que a definição de âncoras também torna os nomes de nó antigos acessíveis quando da leitura do documento em Info.

6.9 `@inforef`: Referências cruzadas para material unicamente Info

`@inforef` é usado para fazer referências cruzadas para documentos Info—mesmo a partir de um manual impresso. Isso pode ser porque você quer referenciar um texto condicional `@ifinfo` (veja Capítulo 16 [Condicionais], Página 128), ou porque a saída impressa não está acessível (talvez porque não exista um fonte Texinfo), entre outras possibilidades.

O comando recebe ou dois ou três argumentos, na seguinte ordem:

1. O nome do nó.
2. O nome da referência cruzada (opcional).
3. O nome do arquivo Info.

O modelo é:

```
@inforef{nome-nó, nome-referência-cruzada, nome-arquivo-info}
```

Por exemplo,

```
@inforef{Avançado, Comandos Info avançados, info},
para mais informação.
```

produz (em Info):

```
*Nota Comandos Info avançados: (info)Avançado,
para mais informação.
```

e (na saída impressa):

> Veja arquivo Info `info`, nó '`Avançado`', para mais informação.

(Esse exemplo em particular não é realístico, dado que o manual Info é escrito em Texinfo, de forma que todos os formatos estão disponíveis. Na verdade, não se sabe de nenhum manual existente unicamente Info).

O inverso de `@inforef` é `@cite`, que é usado para se referir a trabalhos impressos para os quais não existe uma forma Info. Veja Seção 6.11 [`@cite`], Página 55.

6.10 `@url`, `@uref{`*`url`*`[,` *`texto`*`][,` *`substituição`*`]}`

`@uref` produz uma referência a um Localizador Uniforme de Recursos (URL). O comando recebe um argumento obrigatório, a URL, e dois argumentos opcionais que controlam o texto que é exibido. Em saída HTML e PDF, `@uref` produz um link que você pode seguir. (Para meramente indicar uma URL sem criar um link que as pessoas possam seguir, use `@indicateurl`, veja Seção 7.1.15 [`@indicateurl`], Página 63).

`@url` é um sinônimo para `@uref`. (Originalmente, `@url` tinha o significado de `@indicateurl`, porém, na prática, era quase sempre mal usado. Então, mudou-se o significado). O segundo

argumento, se especificado, é o texto a exibir (o padrão é a própria URL); em saída Info, DVI e PDF, porém não em saída HTML, a URL é exibida adicionalmente a esse texto.

O terceiro argumento, se especificado, é o texto a exibir, mas, nesse caso, a URL não é exibida em qualquer formato de saída. Isso é útil quando o texto já está suficientemente referencial, como em uma página de manual. Também, se o terceiro argumento for dado, o segundo argumento é ignorado.

6.10.1 Exemplos `@url`

Primeiro, aqui está um exemplo da forma mais simples de `@url`, com apenas um argumento. A URL dada é tanto o alvo quanto o texto visível do link:

```
O site ftp oficial do projeto GNU é @uref{http://ftp.gnu.org/gnu}.
```

produz:

O site ftp oficial do projeto GNU é `http://ftp.gnu.org/gnu`.

Forma de dois argumentos de `@url`

Aqui está um exemplo da forma de dois argumentos:

```
O @uref{http://ftp.gnu.org/gnu, site ftp oficial do projeto GNU}
contém programas e textos.
```

o que produz:

O site ftp oficial do projeto GNU (`http://ftp.gnu.org/gnu`) contém programas e textos.

isto é, a saída Info (e TEX, etc.) é esta:

```
O site ftp oficial do projeto GNU (http://ftp.gnu.org/gnu) contém
programas e textos.
```

enquanto que a saída HTML é esta:

```
O <a href="http://ftp.gnu.org/gnu">O site ftp oficial do projeto GNU</a>
contém programas e textos.
```

Forma de três argumentos de `@url`

Finalmente, um exemplo da forma de três argumentos:

```
O programa @uref{/man.cgi/1/ls,,ls} ...
```

o qual, exceto para HTML, produz:

O programa ls ...

porém, com HTML:

```
O programa <a href="/man.cgi/1/ls">ls</a> ...
```

A propósito, algumas pessoas preferem exibir URLs no inequívoco formato:

<URL:http://*host*/*path*>

Você pode usar essa forma no arquivo de entrada de desejar. Sentimos que não é necessário incluir o '`<URL:`' e '`>`' na saída, dado que, para ser útil, qualquer software que tente detectar URLs em texto já tem de detectá-las sem o '`<URL:`'.

6.10.2 Quebra de Linha de URL

TEX permite quebra de linha dentro de URLs com apenas alguns caracteres (os quais são especiais em URLs): '`&`', '`.`', '`#`', '`?`', e '`/`', (mas não entre dois caracteres '`/`'). Uma pequena quantidade de espaço elástico também é inserida em torno desses caracteres para ajudar na quebra de linha.

Para saída HTML, os navegadores de Internet modernos também fazem quebra de linha dentro das URLs exibidas. Se você precisar permitir quebras em outros caracteres, você pode inserir `@/` conforme necessário (veja Seção 13.2 [Quebras de Linha], Página 110).

Por padrão, em TEX, quaisquer de tais quebras em caracteres especiais ocorrerão após o caractere. Algumas pessoas preferem que tais quebras aconteçam antes do caractere especial. Isso pode ser controlado com o comando `@urefbreakstyle` (esse comando tem efeito somente em TEX):

```
@urefbreakstyle como
```

onde o argumento *como* é uma destas palavras:

'`after`' (O padrão). Potencialmente quebra após os caracteres especiais.

'`before`' Potencialmente quebra antes dos caracteres especiais.

'`none`' Não considera a quebra em caracteres especiais em absoluto; quaisquer quebras potenciais devem ser inseridas manualmente.

6.10.3 `@url` Formato de Saída PDF

Se o propósito final de um PDF é apenas para ser visto online, talvez similar a HTML de alguma maneira incipiente, você pode não querer que as URLs sejam incluídas no texto visível (exatamente como as URLs não são visíveis a leitores de páginas da web). Texinfo provê uma opção específica de PDF para isso, a qual deve ser usado dentro de `@tex`:

```
@tex
\global\urefurlonlylinktrue
@end tex
```

O resultado é que `@url{http://www.gnu.org, GNU}` tem a saída visível de apenas 'GNU', com um alvo de link de `http://www.gnu.org`. Normalmente, a saída visível incluiria tanto o rótulo quanto a URL: 'GNU (`http://www.gnu.org`)'.

Essa opção somente tem efeito quando a saída PDF é produzida com o programa pdfTEX, não com outras maneiras de obter de Texinfo para PDF (por exemplo, TEX para DVI para PDF). Consequentemente, está certo especificar essa opção incondicionalmente dentro de `@tex`, conforme mostrado acima. A opção é ignorada quando DVI está sendo produzido.

6.10.4 Cores do PDF

Por padrão, links de URLs e de referências cruzadas são impressos em preto em saída PDF. Muito ocasionalmente, entretanto, você pode querer destacar tais links "vivos" com uma cor diferente, como é feito comumente em páginas da web. Texinfo prove uma opção específica de PDF para especificar tais cores, a qual deve ser usada dentro de `@tex`:

```
@tex
\global\def\linkcolor{1 0 0}  % vermelha
\global\def\urlcolor{0 1 0}   % verde
@end tex
```

`\urlcolor` muda a cor da saída `@url` (tanto a URL atual, quanto qualquer rótulo textual), ao passo que `\linkcolor` muda a cor para referências cruzadas para nós, etc. As opções são independentes.

Os três valores dados devem ser números entre 0 e 1, especificando a quantidade de vermelho, verde e azul, respectivamente.

Essas definições somente tem efeito quando a saída PDF é produzida com o programa pdfTEX, não com outros meio de obter de Texinfo para PDF (por exemplo, TEX para DVI para PDF). Consequentemente, está certo especificar essa opção incondicionalmente dentro de `@tex`, conforme mostrado acima. A opção é ignorada quando DVI está sendo produzido.

Não recomendamos colorir apenas por diversão; a menos que você tenha um motivo específico para usar cores, melhor ignorá-las.

6.11 `@cite{`*referência*`}`

Use o comando `@cite` para o nome de um livro que não tenha um arquivo complementar Info. O comando produz itálicos no manual impresso e aspas no arquivo Info.

Se um livro for escrito em Texinfo, é melhor usar um comando de referência cruzada dado que um leitor pode facilmente seguir tal referência em Info. Veja Seção 6.4 [`@xref`], Página 47.

7 Marcação de Texto, Palavras e Frases

Em Texinfo, você pode marcar palavras e frases de várias formas. Os formatadores Texinfo usam essa informação para determinar como destacar o texto. Você pode especificar, por exemplo, quando uma palavra ou frase é uma ocorrência definidora, uma variável metassintática ou um símbolo usado em um programa. Além disso, você pode enfatizar o texto, de diferentes maneiras.

7.1 Indicando Definições, Comandos, etc.

Texinfo tem comandos para indicar exatamente a que tipo de objeto um pedaço de texto se refere. Por exemplo, endereços de correio eletrônico são marcados por `@email`; dessa maneira, o resultado pode ser um link "vivo" para enviar mensagem eletrônica quando o formato de saída for compatível. Se o endereço de correio eletrônico simplesmente fosse marcado como "imprimir em uma fonte de máquina de escrever", isso não seria possível.

7.1.1 Comandos de Realçamento são Úteis

Os comandos servem a uma variedade de propósitos:

`@code{`*código-modelo*`}`
: Indica texto que é um exemplo literal de um pedaço de um programa. Veja Seção 7.1.2 [`@code`], Página 57.

`@kbd{`*caracteres-de-teclado*`}`
: Indica entrada de teclado. Veja Seção 7.1.3 [`@kbd`], Página 58.

`@key{`*nome-de-tecla*`}`
: Indica o nome convencional para uma tecla em um teclado. Veja Seção 7.1.4 [`@key`], Página 59.

`@samp{`*texto*`}`
: Indica texto que é um exemplo literal de uma sequência de caracteres. Veja Seção 7.1.5 [`@samp`], Página 59.

`@verb{`*texto*`}`
: Escreve uma sequência literal de caracteres. Veja Seção 7.1.6 [`@verb`], Página 60.

`@var{`*variável-metassintática*`}`
: Indica uma variável metassintática. Veja Seção 7.1.7 [`@var`], Página 60.

`@env{`*variável-de-ambiente*`}`
: Indica uma variável de ambiente. Veja Seção 7.1.8 [`@env`], Página 61.

`@file{`*nome-de-arquivo*`}`
: Indica o nome de um arquivo. Veja Seção 7.1.9 [`@file`], Página 61.

`@command{`*nome-de-comando*`}`
: Indica o nome de um comando. Veja Seção 7.1.10 [`@command`], Página 61.

`@option{`*nome-de-opção*`}`
: Indica uma opção de linha de comando. Veja Seção 7.1.11 [`@option`], Página 62.

`@dfn{`*termo*`}`
: Indica o uso definidor ou introdutório de um termo. Veja Seção 7.1.12 [`@dfn`], Página 62.

`@cite{`*referência*`}`
: Indica o nome de um livro. Veja Seção 6.11 [`@cite`], Página 55.

`@abbr{`*abreviação*`}`
: Indica uma abreviação, como 'Comput.'.

`@acronym{`*`sigla`*`}`
: Indica uma sigla. Veja Seção 7.1.14 [`@acronym`], Página 62.

`@indicateurl{`*`localizador-uniforme-de-recurso`*`}`
: Indica um exemplo (isto é, não funcional) de Localizador Uniforme de Recurso. Veja Seção 7.1.15 [`@indicateurl`], Página 63. (Use `@url` (veja Seção 6.10 [`@url`], Página 52) para URLs "vivas").

`@email{`*`endereço-de-email`*`[, `*`texto-exibido`*`]}`
: Indica um endereço de correio eletrônico. Veja Seção 7.1.16 [`@email`], Página 63.

7.1.2 `@code{`*`código-modelo`*`}`

Use o comando `@code` para indicar texto que é um pedaço de um programa e que consiste de símbolos sintáticos inteiros. Coloque o texto entre chaves.

Assim, você deveria usar `@code` para uma expressão em um programa, para o nome de uma variável ou função usada em um programa ou para uma palavra chave em uma linguagem de programação.

Use `@code` para nomes de comando em linguagens que reconstroem linguagens de programação, como Texinfo. Por exemplo, `@code` e `@samp` são produzidos escrevendo '`@code{@@code}`' e '`@code{@@samp}`' no fonte Texinfo, respectivamente.

É incorreto alterar a primeira letra (para maiúscula) de uma palavra dentro de um comando `@code` quando esse aparece no início de uma frase. A maioria das linguagens de programação diferencia maiúsculas de minúsculas. Em C, por exemplo, `Printf` é diferente do identificador `printf`, e possivelmente é um erro de ortografia. Mesmo em linguagens que não diferenciam maiúsculas de minúsculas, é confuso para um leitor humano ver identificadores escritos de diferentes maneiras. Escolha uma grafia e sempre a use. Se você não quer iniciar uma frase com um nome de comando escrito todo em letras minúsculas, você deveria reorganizar a frase.

Na saída Info, `@code` resulta em aspas simples em torno do texto. Em outros formatos, o argumento `@code` é formatado em uma fonte de máquina de escrever (mono espaçada). Por exemplo,

```
A função retorna @code{nulo}.
```

produz isto:

A função retorna `nulo`.

Aqui estão alguns casos para os quais é preferível *não* usar `@code`:

- Para nomes de comando de shell como `ls` (use `@command`).
- Para variável de ambiente como `TEXINPUTS` (use `@env`).
- Para opções de shell como '`-c`' quando tais opções são independentes (use `@option`).
- Um comando inteiro de shell frequentemente parece melhor se escrito usando `@samp` em vez de `@code`. Nesse caso, a regra é escolher o formato mais agradável.
- Para uma sequência de caracteres menor que um token sintático. Por exemplo, se você está escrevendo sobre '`goto-ch`', que apenas é parte do nome para a função `goto-char` da Emacs Lisp, você deveria usar `@samp`.
- Geralmente, quando escrever acerca dos caracteres usados em um token; por exemplo, não use `@code` quando estiver explanando quais letras ou símbolos imprimíveis podem ser usados nos nomes das funções. (Use `@samp`). Ainda, você deveria não usar `@code` para marcar texto que é considerado entrada para programas, a menos que a entrada seja escrita em uma linguagem que seja semelhante a uma linguagem de programação. Por exemplo, você não deveria usar `@code` para os comandos de teclas do GNU Emacs (use `@kbd` em vez disso), apesar que você pode usar `@code` para os nomes das funções da Emacs Lisp que os comandos de teclas invocam.

Por padrão, TEX considerará quebras de linhas nos caracteres '-' e '_' dentro do `@code` e comandos relacionados. Isso pode ser controlado com `@allowcodebreaks` (veja Seção 13.4 [`@allowcodebreaks`], Página 111). A saída HTML tenta respeitar isso para '-', mas, no final, depende do comportamento do navegador. Para Info, parece melhor nunca fazer tais quebras.

Para Info, as aspas são omitidas na saída do comando `@code` e comandos relacionados (por exemplo, `@kbd`, `@command`), em contextos semelhantes a máquinas de escrever, como o ambiente `@example` (veja Seção 8.4 [`@example`], Página 68) e o próprio `@code`, etc.

Para controlar quais caracteres de citação são implicitamente inseridos por processadores Texinfo na saída do '`@code`', etc., veja as variáveis de personalização `OPEN_QUOTE_SYMBOL` e `CLOSE_QUOTE_SYMBOL` (veja Seção 20.5.4 [Outras Variáveis de Personalização], Página 177). Isso é separado de como são manipulados os atuais caracteres de citação no documento de entrada (veja Seção 12.2 [Inserindo Caracteres de Citação], Página 97).

7.1.3 `@kbd{`*caracteres-de-teclado*`}`

Use o comando `@kbd` para caracteres de entrada a serem digitados por usuários. Por exemplo, para se referir aos caracteres *`M-a`*, escreva:

```
@kbd{M-a}
```

e para se referir aos caracteres *`M-x shell`*, escreva:

```
@kbd{M-x shell}
```

Por padrão, o comando `@kbd` produz uma fonte diferente (máquina de escrever inclinada em vez da máquina de escrever normal), de forma que os usuários podem distinguir os caracteres que deveriam digitar daqueles que o computador exibe.

Dado que o uso de `@kbd` varia de manual para manual, você pode controlar a fonte trocando com o comando `@kbdinputstyle`. Esse comando não tem efeito sobre a saída Info. Escreva esse comando no início de uma linha com uma única palavra como um argumento, uma das seguintes:

'`code`' Sempre use a mesma fonte para `@kbd` como `@code`.

'`example`' Use a fonte de distinção para `@kbd` somente em `@example` e ambientes similares.

'`distinct`'
(o padrão) Sempre use a fonte de distinção para `@kbd`.

Você pode incorporar outro comando @ dentro das chaves de um comando `@kbd`. Aqui, por exemplo, está a maneira para descrever um comando que pressiona a tecla `RETURN`":

```
@kbd{r @key{RET}}
```

Isso produz: *`r RET`*. (O presente manual usa o padrão para `@kbdinputstyle`).

Você também usa o comando `@kbd` se estiver soletrando as letras que digitar; por exemplo:

```
Para dar o comando @code{logout}, digite os caracteres
@kbd{l o g o u t @key{RET}}.
```

Isso produz:

> Para dar o comando `logout`, digite os caracteres *`l o g o u t RET`*.

(Além disso, esse exemplo mostra que você pode adicionar espaços para maior clareza. Se você deseja mencionar explicitamente um caractere espaço como um dos caracteres da entrada, escreva *`@key{SPC}`* para ele).

7.1.4 @key{*nome-de-tecla*}

Use o comando `@key` para o nome convencional para uma tecla em um teclado, como em:

```
@key{RET}
```

Você pode usar o comando `@key` dentro do argumento de um comando `@kbd` quando a sequência de caracteres a ser digitada incluir uma ou mais teclas que sejam descritas por nome.

Por exemplo, para produzir *C-x ESC* e *M-TAB* você digitaria:

```
@kbd{C-x @key{ESC}}
@kbd{M-@key{TAB}}
```

Aqui está uma lista dos nomes recomendados para teclas:

`SPC`	Espaço
`RET`	Retorno (Nova linha)
`LFD`	Alimentação de Linha (no entanto, como a maioria dos teclados hoje em dia não tem uma tecla de alimentação de linha, talvez seja melhor chamar esse caractere *C-j*)
`TAB`	Tab
`BS`	Backspace
`ESC`	Escape
`DELETE`	Delete
`SHIFT`	Shift
`CTRL`	Control
`META`	Meta

Existem sutilezas para lidar com palavras como 'meta' ou 'ctrl', que são nomes de teclas modificadoras. Quando mencionar um caractere no qual a tecla modificadora é usada, como *Meta-a*, use o comando `@kbd` sozinho; não use o comando `@key`; porém, quando estiver se referindo à tecla modificadora isoladamente, use o comando `@key`. Por exemplo, escreva '`@kbd{Meta-a}`' para produzir *Meta-a* e '`@key{META}`' para produzir `META`.

Como uma convenção em manuais GNU, `@key` não deveria ser usado em entradas de indexação.

7.1.5 @samp{*texto*}

Use o comando `@samp` para indicar um texto que seja um exemplo literal ou "amostra" de uma sequência de caracteres em um arquivo, cadeia de caracteres, padrão, etc. Coloque o texto entre chaves. O argumento aparece entre aspas simples tanto no arquivo Info quanto no manual impresso; além disso, o texto é impresso em uma fonte de largura fixa.

```
Para combinar @samp{foo} no final da linha, use a expressão regular
@samp{foo$}.
```

produz:

Para combinar '`foo`' no final da linha, use a expressão regular '`foo$`'.

Sempre que estiver se referindo a caracteres únicos, você deveria usar `@samp`, a menos que `@kbd` ou `@key` sejam mais apropriados. Além disso, você pode usar `@samp` para instruções inteiras em C e para comandos inteiros do shell—nesse caso, `@samp` geralmente parece melhor que `@code`. Basicamente, `@samp` é um "pega tudo" para o que não for coberto por `@code`, `@kbd`, `@key`, `@command`, etc.

Somente inclua sinais de pontuação entre chaves se eles fizerem parte da sequência de caracteres que você estiver especificando. Escreva sinais de pontuação fora das chaves se tais sinais de pontuação fizerem parte do texto em Inglês que envolve a sequência de caracteres. Na frase seguinte, por exemplo, as vírgulas e pontos estão fora das chaves:

```
Em Inglês, as vogais são @samp{a}, @samp{e}, @samp{i},
@samp{o}, @samp{u}, e, as vezes, @samp{y}.
```

Isso produz:

Em Inglês, as vogais são ‘`a`’, ‘`e`’, ‘`i`’, ‘`o`’, ‘`u`’, e, as vezes, ‘`y`’.

7.1.6 `@verb{`*chartextchar*`}`

Use o comando `@verb` para imprimir uma sequência literal de caracteres.

Semelhante ao comando `\verb` do LaTeX, o texto literal pode ser delimitado usando-se qualquer caractere delimitador único. Coloque o texto literal, incluindo os delimitadores, entre chaves. O texto é impresso em uma fonte de largura fixa:

```
Quantos caracteres de escape @verb{|@|} alguém precisa para imprimir
esta sequência @verb{.@a @b.@c.} ou esta
@verb{+@'e?`{}!`\+}?
```

produz:

```
Quantos caracteres de escape @ alguém precisa para imprimir
esta sequência @a @b.@c ou esta @'e?`{}!`\?
```

Isso está em contraste com `@samp` (veja a seção anterior), `@code` e comandos semelhantes; naqueles casos, o argumento é texto normal Texinfo, onde os três caracteres `@{}` são especiais, como de costume. Com `@verb`, nada é especial, exceto o caractere delimitador que você escolher.

O próprio caractere delimitador pode aparecer dentro do texto literal, conforme mostrado acima. Como outro exemplo, ‘`@verb{...}`’ imprime um único ponto (largura fixa).

Não é confiável usar `@verb` dentro de outras construções do Texinfo. Em particular, não funciona usar `@verb` em nada relacionado a referências cruzadas, como títulos de seção ou legendas de figuras.

7.1.7 `@var{`*variável-metassintática*`}`

Use o comando `@var` para indicar variáveis metassintáticas. Uma *variável metassintática* é algo que representa outro pedaço de texto. Por exemplo, você deveria usar uma variável metassintática na documentação de uma função para descrever os argumentos que são passados para aquela função.

Não use `@var` para os nomes de variáveis normais em programas de computador. Esse são nomes específicos, de forma que `@code` é correto para eles (`@code`). Por exemplo, a variável `texinfo-tex-command` da Emacs Lisp não é uma variável metassintática; ela é formatada apropriadamente usando `@code`.

Tampouco use `@var` para variáveis de ambiente; `@env` é correto para elas (veja a próxima seção).

O efeito de `@var` no arquivo Info é modificar a escrita do argumento para letras todas maiúsculas. No manual impresso e saída HTML, o argumento é exibido em tipo inclinado.

Por exemplo,

```
Para deletar o arquivo @var{filename}, digite
@samp{rm @var{filename}}.
```

produz:

Para deletar o arquivo *filename*, digite ‘`rm` *`filename`*’.

(Note que `@var` pode aparecer dentro de `@code`, `@samp`, `@file`, etc.).

Escreva uma variável metassintática toda em letras minúsculas e sem espaços, e use hifens para torná-la mais legível. Assim, o fonte de Texinfo para a ilustração de como iniciar um manual Texinfo se parece com isto:

```
\input texinfo
@@settitle @var{nome-do-manual}
```

Isso produz:

```
\input texinfo
@settitle nome-do-manual
```

Em alguns estilos de documentação, as variáveis metassintáticas são mostradas entre colchetes angulares, por exemplo:

```
..., digite rm <nome_arquivo>
```

Entretanto, esse não é o estilo que Texinfo usa.

7.1.8 `@env{`*variável-de-ambiente*`}`

Use o comando `@env` para indicar variáveis de ambiente, conforme usadas por muitos Sistemas Operacionais, incluindo GNU. Não a use para variáveis *meta*ssintáticas; use `@var` para essas (veja a seção anterior).

`@env` é equivalente a `@code` em seus efeitos. Por exemplo:

```
A variável de ambiente @env{PATH} ...
```

produz:

A variável de ambiente `PATH` ...

7.1.9 `@file{`*nome-de-arquivo*`}`

Use o comando `@file` para indicar texto que é o nome de um arquivo, buffer, ou diretório, ou é o nome de um nó em Info. Você também pode usar o comando para extensões de nome de arquivo. Não use `@file` para símbolos em uma linguagem de programação; use `@code`.

`@file` é equivalente a `code` em seus efeitos. Por exemplo,

```
Os arquivos @file{.el} estão no diretório
@file{/usr/local/emacs/lisp}.
```

produz:

Os arquivos `.el` estão no diretório `/usr/local/emacs/lisp`.

7.1.10 `@command{`*nome-de-comando*`}`

Use o comando `@command` para indicar os nomes de comando, como `ls` ou `cc`.

`@command` é equivalente a `@code` em seus efeitos. Por exemplo:

```
O comando @command{ls} lista o conteúdo de diretórios.
```

produz:

O comando `ls` lista o conteúdo de diretórios.

Você deveria escrever o nome de um programa na fonte de texto comum, em vez de usar `@command`, se considerá-lo uma nova palavra em inglês, como ‘Emacs’ ou ‘Bison’.

Quando escrever uma invocação inteira de comando de shell, como em ‘`ls -l`’, você deveria usar ou `@samp` ou `@code` a seu critério.

7.1.11 `@option{`*nome-de-opção*`}`

Use o comando `@option` para indicar uma opção de linha de comando; por exemplo, `-l` ou `--version` ou `--output=`*`filename`*.

`@option` é equivalente a `@code` em seus efeitos. Por exemplo:

```
A opção @option{-l} produz uma listagem longa.
```

produz:

A opção `-l` produz uma listagem longa.

7.1.12 `@dfn{`*termo*`}`

Use o comando `@dfn` para identificar o uso introdutório ou definidor de um termo técnico. Use o comando somente em passagens cujo propósito é introduzir um termo o qual será usado novamente ou o qual o leitor deve saber. A simples menção de um termo pela primeira vez não merece um `@dfn`. O comando gera itálicos no manual impresso, e aspas duplas no arquivo Info. Por exemplo:

```
Livrar-se de um arquivo chama-se @dfn{deletando}.
```

produz:

Livrar-se de um arquivo chama-se *deletando*.

Como regra geral, uma frase contendo a ocorrência definidora de um termo deve ser uma definição do termo. A frase não precisa dizer explicitamente que é uma definição, mas deve conter a informação de uma definição—ela deve deixar claro o significado.

7.1.13 `@abbr{`*abreviação*`[, `*significado*`]}`

Você pode usar o comando `@abbr` para abreviações gerais. A abreviação é dada como o argumento único entre chaves, como em '`@abbr{Comput.}`'. Por uma questão de estilo, ou para abreviações específicas, você pode preferir omitir pontos, como em '`@abbr{Mr} Stallman`'.

`@abbr` aceita um segundo argumento opcional, destinado a ser usado para o significado da abreviação.

Se a abreviação finalizar com uma letra minúscula e um ponto, e não estiver no final de uma frase, e não tiver um segundo argumento, lembre-se de usar o comando `@.` (veja Seção 12.3.3 [Finalizando Uma Frase], Página 98) para obter o espaçamento correto. Entretanto, você não tem de usar `@.` dentro da própria abreviação; Texinfo automaticamente assume que pontos dentro da abreviação não finalizam uma frase.

Em TEX e na saída Info, o primeiro argumento é impresso como está; se o segundo argumento estiver presente, ele é impresso entre parênteses após a abreviação. Em HTML, a marcação `<abbr>` é usada; em Docbook, a marcação `<abbrev>` é usada. Por exemplo:

```
@abbr{Comput. J., Jornal do Computador}
```

produz:

Comput. J. (Jornal do Computador)

Para abreviações que consistem de todas as letras maiúsculas, você pode preferir usar o comando `@acronym`. Veja a próxima seção para mais sobre o uso desses dois comandos.

7.1.14 `@acronym{`*sigla*`[, `*significado*`]}`

Você pode usar o comando `@acronym` para abreviações escritas todas com letras maiúsculas, como 'NASA'. A abreviação é dada como o único argumento entre chaves, como em '`@acronym{NASA}`'. Por uma questão de estilo, ou para siglas específicas, você pode preferir usar pontos, como em '`@acronym{N.A.S.A.}`'.

`@acronym` aceita um segundo argumento opcional, destinado a ser usado para o significado da sigla.

Se a sigla estiver no final de uma frase, e se não existir segundo argumento, lembre-se de usar o `@.` ou comando similar (veja Seção 12.3.3 [Finalizando Uma Frase], Página 98) para obter o espaçamento correto.

Em TEX, a sigla é impressa em fonte ligeiramente menor. Na saída Info, o argumento é impresso como está. Em qualquer formato, se o segundo argumento estiver presente, ele é impresso entre parênteses após a sigla. Em HTML e Docbook a marcação `<acronym>` é usada.

Por exemplo (dado que GNU é uma sigla recursiva, nós usamos `@acronym` recursivamente):

```
@acronym{GNU, @acronym{GNU} Não é Unix}
```

produz:

GNU (GNU Não é Unix)

Em algumas circunstâncias, é convencional imprimir nomes de família em letras maiúsculas. Não use `@acronym` para isso, pois um nome não é uma sigla. Use `@sc` (veja Seção 7.2.2 [Versaletes], Página 64).

`@abbr` e `@acronym` são comandos intimamente relacionados: ambos sinalizam para o(a) leitor(a) que uma forma abreviada está sendo usada, e possivelmente dá um significado. Ao escolher usar esses dois comandos, por favor tenha o seguinte em mente.

- No uso comum do Inglês, siglas são um subconjunto de abreviações: elas incluem palavras pronunciáveis como 'NATO', 'radar' e 'snafu'; algumas fontes também incluem siglas de siglas, como 'Usenet'; híbridas, como 'SIGGRAPH'; e inicialismos impronunciáveis como 'FBI'.
- Em Texinfo, uma sigla (mas não uma abreviação) deveria consistir apenas de letras maiúsculas e pontos, não minúsculas.
- Em TEX, uma sigla (mas não uma abreviação) é impressa em uma fonte ligeiramente menor.
- Alguns navegadores colocam uma borda inferior pontilhada em abreviações, mas não em siglas.
- Geralmente é bastante difícil e/ou demorado usar consistentemente `@acronym` para todas as sequências de letras maiúsculas. Além do mais, parece estranho para algumas siglas estar no tamanho normal da fonte e outras estarem menores. Assim, uma abordagem mais simples que você pode querer considerar é evitar `@acronym` e apenas digitar tudo como texto normal em todas maiúsculas: '`GNU`', produzindo a saída 'GNU'.
- Em geral, não é essencial usar quaisquer desses comandos para todas as abreviações; use seu bom senso. O texto é perfeitamente legível sem tais comandos.

7.1.15 `@indicateurl{`*localizador-uniforme-de-recurso*`}`

Use o comando `@indicateurl` para indicar um Localizador Uniforme de Recurso na Internet. Isso é puramente para propósitos de marcação e não produz um link que você possa seguir (use o comando `@url` ou `@uref` para isso, veja Seção 6.10 [`@url`], Página 52). `@indicateurl` é útil para URLs que não existem atualmente. Por exemplo:

```
Por exemplo, a URL poderia ser @indicateurl{http://example.org/path}.
```

que produz:

Por exemplo, a URL poderia ser '`http://example.org/path`'.

A saída oriunda de `@indicateurl` é mais ou menos como aquela de `@samp` (veja Seção 7.1.5 [`@samp`], Página 59).

7.1.16 `@email{`*endereço-de-email*`[, `*texto-exibido*`]}`

Use o comando `@email` para indicar um endereço de correio eletrônico. O comando recebe um argumento obrigatório, o endereço, e um argumento opcional, o texto a exibir (o padrão é o próprio endereço).

Em Info, o endereço é mostrado entre colchetes angulares (sinal de "maior que" e "menor que"), precedido pelo texto a ser exibido, se houver. Em TEX, os colchetes angulares são omitidos. Na saída HTML, `@email` produz um link '`mailto`' que geralmente abre uma janela de composição de correspondência. Por exemplo:

```
Mande relatórios de bugs para @email{bug-texinfo@@gnu.org},
sugestões para o @email{bug-texinfo@@gnu.org, algum lugar}.
```

produz:

> Mande relatórios de bugs para `bug-texinfo@gnu.org`,
> sugestões para o algum lugar.

7.2 Enfatizando Texto

Normalmente, Texinfo modifica a fonte para marcar palavras no texto de acordo com a categoria a que as palavras pertencem; um exemplo é o comando `@code`. Na maioria das vezes, essa é a melhor maneira de marcar palavras. Entretanto, as vezes, você desejará enfatizar texto sem indicar uma categoria. Texinfo tem dois comandos para fazer isso. Além disso, Texinfo tem vários comandos que especificam a fonte na qual o texto será produzido. Esses comandos não tem efeito em Info e apenas um deles, o comando `@r`, tem algum uso regular.

7.2.1 `@emph{`*texto*`}` e `@strong{`*texto*`}`

Os comandos `@emph` e `@strong` são para ênfase; `@strong` é o mais forte. Em saída impressa, `@emph` produz *itálico* e `@strong` produz **negrito**. Na saída Info, `@emph` envolve o texto com sublinhados ('_'), e `@strong` coloca asteriscos em volta do texto.

Por exemplo,

```
@strong{Cuidado:} @samp{rm *} remove @emph{todos} os arquivos normais.
```

produz o seguinte:

> **Cuidado**: '`rm * .[^.]*`' remove *todos* os arquivos normais.

O comando `@strong` raramente é usado, exceto para marcar que é, na verdade, um elemento tipográfico, como a palavra 'Cuidado' no exemplo anterior.

> **Cuidado:** Não use `@strong` com a palavra '`Nota`' seguida por um espaço; Info confundirá a combinação para uma referência cruzada. Use uma frase, como **Por favor note** ou **Cuidado** em vez disso, ou o argumento opcional para `@quotation`—'`Nota`' é permitido lá.

7.2.2 `@sc{`*texto*`}`: A Fonte de Versaletes

Use o comando '`@sc`' para configurar texto em UMA FONTE DE VERSALETES (onde for possível). Escreva o texto que você quer que esteja em versaletes entre chaves e em letras minúsculas, como isto:

```
Richard @sc{Stallman} commencé GNU.
```

Isso produz:

> Richard STALLMAN commencé GNU.

Como mostrado aqui, nós recomendamos reservar `@sc` para casos especiais onde você desejar versalete tipográfica; nomes de família são um desses, especialmente em outros idiomas que não o Inglês, embora não existam regras rígidas e rápidas acerca de tais coisas.

TEX escreve qualquer letra maiúscula entre as chaves de um comando `@sc` em letras maiúsculas de tamanho real; apenas letras minúsculas são impressas na fonte de versaletes. Na saída Info, o argumento para `@sc` é impresso todo em maiúsculas. Em HTML, o argumento é colocado em letras maiúsculas e a saída marcada com a marcação `<small>` para reduzir o

tamanho da fonte, dado que HTML não pode facilmente representar capitalizações pequenas verdadeiras.

No geral, recomendamos o uso de letras maiúsculas e minúsculas padrão sempre que possível.

7.2.3 Fontes para Impressão

O Texinfo fornece um comando para mudar o tamanho da fonte do corpo principal na saída TEX para um documento: `@fonttextsize`. O comando não tem efeito em outra saída. O comando recebe um argumento único no restante da linha, o qual deve ser ou '`10`' ou '`11`'. Por exemplo:

```
@fonttextsize 10
```

O efeito é reduzir a fonte do corpo para um tamanho de 10 pt (o padrão é 11 pt). Fontes para outros elementos, tais como seções e capítulos, são reduzidas de acordo. Isso somente deveria ser usado em conjunto com `@smallbook` (veja Seção 19.11 [`@smallbook`], Página 159) ou similar, dado que fontes de 10 pt em papel padrão (8.5x11 ou A4) são muito pequenas. Uma razão para usar esse comando é para economizar páginas, e, assim, custo de impressão, para livros físicos.

Texinfo atualmente não tem comandos para alternar a família da fonte a usar, ou comandos de alteração de tamanho mais gerais.

Texinfo também fornece vários comandos de fonte que especificam as mudanças de fonte no manual impresso e (quando possível) na saída HTML. Esses comandos não tem efeito em Info. Todos os comandos se aplicam a um argumento a seguir cercados por chaves.

`@b` seleciona **negrito**;

`@i` seleciona uma fonte *itálico*;

`@r` seleciona uma fonte romana, que é a fonte usual na qual o texto é impresso. Ela pode ou não ser serifa.

`@sansserif`
seleciona uma fonte sans serifa;

`@slanted` seleciona uma fonte *inclinada*;

`@t` seleciona a fonte de `tamanho fixo`, estilo máquina de escrever usada por `@code`;

(Os comandos com nomes mais longos foram inventados muito mais tarde que os outros, quando não parecia desejável usar nomes muito curtos para recursos tão pouco necessários).

O comando `@r` pode ser útil em ambientes de exemplo, para escrever comentários na fonte romana padrão em vez da fonte de largura fixa. Isso fica melhor na saída impressa, e produz uma marca `<lineannotation>` na saída do Docbook.

Por exemplo,

```
@lisp
(+ 2 2)    ; @r{Adiciona dois mais dois.}
@end lisp
```

produz

```
(+ 2 2)    ; Adiciona dois mais dois.
```

O comando `@t` pode ocasionalmente ser útil para produzir saída em uma fonte de máquina de escrever onde isso for suportado (por exemplo, HTML e PDF), mas nenhuma distinção é necessária em Info ou texto plano: `@t{foo}` produz `foo`, cf. `@code{foo}` produzindo `foo`.

Em geral, os outros comandos de fonte provavelmente não são úteis; eles existem principalmente para possibilitar documentar a funcionalidade de efeitos específicos de fontes, como no TEX e pacotes relacionados.

8 Citações e Exemplos

Citações e exemplos são blocos de texto que consistem em um ou mais parágrafos inteiros que são separados da maior parte do texto e tratados de forma diferente. Eles geralmente são recuados na saída.

Em Texinfo, você sempre inicia uma citação ou exemplo escrevendo um comando @ sozinho no início de uma linha, e o finaliza escrevendo um comando `@end` que também está sozinho no início de uma linha. Por exemplo, você inicia um exemplo escrevendo `@example` no início de uma linha e finaliza o exemplo escrevendo `@end example` sozinho, no início daquela linha, e somente com um espaço entre o `@end` e o `example`.

8.1 Comandos de Cercamento de Blocos

Aqui está um resumo de comandos que contém blocos de texto, também conhecidos como *ambientes*. Eles estão mais explicados nas próximas seções.

`@quotation`
: Indica texto que é citado. O texto é preenchido, recuado (a partir de ambas as margens) , e impresso em uma fonte romana por padrão.

`@indentedblock`
: Como `@quotation`, mas o texto é recuado apenas a esquerda.

`@example`
: Ilustra código, comandos, e afins. O texto é impresso em uma fonte de largura fixa, e recuado, mas não preenchido.

`@lisp`
: Como `@example`, mas especificamente para ilustrar código da Lisp. O texto é impresso em uma fonte de largura fixa, e recuado, mas não preenchido.

`@verbatim`
: Marca um pedaço de texto que é para ser impresso literalmente; nenhuma substituição de caractere é feita e todos os comandos são ignorados, até o próximo `@end verbatim`. O texto é impresso em uma fonte de largura fixa, e não é recuado ou preenchido. Espaços extra e linhas em branco são significantes, e TABs são expandidos.

`@display`
: Exibe texto ilustrativo. O texto é recuado, mas não preenchido, e nenhuma fonte é selecionada (então, por padrão, a fonte ŕ romana).

`@format`
: Como `@display` (o texto não é preenchido e nenhuma fonte é selecionada), mas o texto não é recuado.

`@smallquotation`
`@smallindentedblock`
`@smallexample`
`@smalllisp`
`@smalldisplay`
`@smallformat`
: Esses comandos `@small...` são semelhantes aos seus equivalentes não-small, exceto que emitem texto em um tamanho de fonte menor, quando possível.

`@flushleft`
`@flushright`
: O texto não é preenchido, mas é alinhado com a margem esquerda ou direita, respectivamente.

`@raggedright`
: O texto é preenchido, mas justificado apenas a esquerda, deixando a margem direita irregular.

`@cartouche`
: Realça o texto, geralmente um exemplo ou uma citação, desenhando uma caixa com cantos arredondados ao redor.

O comando `@exdent` é usado dentro das construções acima para desfazer o recuo de uma linha.

O comando `@noindent` pode ser usado após uma das construções acima (ou no início de qualquer parágrafo) para prevenir que o texto seguinte seja recuado como um novo parágrafo.

8.2 `@quotation`: Citações de Bloco

O texto de uma citação é processado como texto normal (fonte regular, texto é preenchido), exceto que:

- ambas as margens esquerda e direita estão mais próximas ao centro da página, de forma que a totalidade da citação é recuada;
- as primeiras linhas de parágrafos não são recuadas mais que outras linhas; e
- um comando `@author` pode ser dado para especificar o autor da citação.

> Este é um exemplo de texto escrito entre um comando `@quotation` e um comando `@end quotation`. Um comando `@quotation` é usado com frequência para indicar texto que é extraído de outro trabalho impresso (real ou hipotético).

Escreva um comando `@quotation` como texto sozinho em uma linha. Essa linha desaparecerá da saída. Marque o final da citação com uma linha iniciando com e contendo somente `@end quotation`. A linha do `@end quotation` também desaparecerá da saída.

`@quotation` aceita um argumento opcional, dado no restante da linha. Esse texto, se presente, é incluído no início da citação, em negrito, ou, caso contrário, enfatizado, e seguido com um '`:`'. Por exemplo:

```
@quotation Nota
Isto é
um foo.
@end quotation
```

produz:

> **Nota:** Isto é um foo.

Se o argumento do `@quotation` for uma destas palavras do Inglês (não diferencia maiúsculas de minúsculas):

```
Caution  Important  Note  Tip  Warning
```

então a saída do Docbook usa marcações especiais correspondentes (`<note>`, etc.) em vez do padrão `<blockquote>`. A saída HTML sempre usa `<blockquote>`.

Se o autor da citação for especificado no bloco `@quotation` com o comando `@author`, uma linha com o nome do autor é exibida após a citação:

```
@quotation

As pessoas as vezes me perguntam se é pecado na Igreja do Emacs usar vi.
Usar uma versão livre do vi não é pecado; é uma penitência. Então, feliz
hacking.

@author Richard Stallman
@end quotation
```

produz

> As pessoas as vezes me perguntam se é pecado na Igreja do Emacs usar vi. Usar uma versão livre do vi não é pecado; é uma penitência. Então, feliz hacking.
> —*Richard Stallman*

Texinfo também fornece um comando `@smallquotation`, o qual é exatamente como `@quotation`, mas usa um tamanho de fonte menor onde for possível. Veja Seção 8.15 [`@small...`], Página 73.

8.3 `@indentedblock`: Blocos recuados de texto

O ambiente `@indentedblock` é semelhante ao `@quotation`, exceto que o texto somente é recuado a esquerda (e não existe argumento opcional para um autor). Assim, a fonte do texto permanece inalterada, e o texto é reunido e preenchido como de costume, mas a margem esquerda é aumentada. Por exemplo:

> Este é um exemplo de texto escrito entre um comando `@indentedblock` e um comando `@end indentedblock`. O ambiente `@indentedblock` pode conter qualquer texto ou outros comandos desejados.

Isso é escrito no fonte do Texinfo como:

```
@indentedblock
Este é um exemplo ...
@end indentedblock
```

Texinfo também fornece um comando `@smallindentedblock`, que é exatamente como `@indentedblock`, mas usa um tamanho de fonte menor onde possível. Veja Seção 8.15 [`@small...`], Página 73.

8.4 `@example`: Texto de Exemplo

O ambiente `@example` é usado para indicar um exemplo que não é parte do texto em execução, como uma saída ou entrada de computador. Escreva um comando `@example` sozinho no início de uma linha. Marque o final do exemplo com um comando `@end example`, também escrito sozinho no início de uma linha.

Um ambiente `@example` tem as seguintes características:

- Cada linha no arquivo de entrada é uma linha na saída; isto é, o texto fonte não é preenchido como normalmente é.
- Linhas em branco e espaços extras são significantes.
- A saída é recuada.
- A saída usa uma fonte de largura fixa.
- Os comandos do Texinfo *são* expandidos; se você deseja que a saída seja a entrada literalmente, use o ambiente `@verbatim` (veja Seção 8.5 [`@verbatim`], Página 69).

Por exemplo,

```
@example
cp foo @var{dest1}; \
 cp foo @var{dest2}
@end example
```

produz:

```
cp foo dest1; \
 cp foo dest2
```

As linhas contendo `@example` e `@end example` desaparecerão da entrada. Para fazer com que a saída apareça bem, você deveria colocar uma linha em branco antes do `@example` e outra linha

em branco após o `@end example`. As linhas em branco dentro do `@example` inicial e do `@end example` final, por outro lado, aparecem na saída.

> **Cuidado:** Não use "tabs" nas linhas de um exemplo! (Ou em qualquer outro lugar em Texinfo, exceto em ambientes "verbatim"). O TEX trata os "tabs" como espaços simples, e isso não é o que parecem. Em Emacs, você pode usar *`M-x untabify`* para converter "tabs" em uma região para espaços múltiplos.

Os exemplos frequentemente estão, falando logicamente, "no meio" de um parágrafo, e o texto que continua após não deveria ser recuado, como no exemplo acima. O comando `@noindent` evita que um pedaço de texto seja recuado como se fosse um parágrafo novo (veja Seção 8.12 [`@noindent`], Página 72.

Se você deseja incorporar fragmentos de código em frases, em vez de exibi-los, use o comando `@code` ou seus parentes (veja Seção 7.1.2 [`@code`], Página 57).

Se você quer escrever um "comentário" em uma linha de um exemplo na fonte romana normal, você pode usar o comando `@r` (veja Seção 7.2.3 [Fontes], Página 65).

8.5 `@verbatim`: Texto Literal

Use o ambiente `@verbatim` para a impressão de texto que pode conter caracteres especiais ou comandos que não deveriam ser interpretados, como saída ou entrada de computador (`@example` interpreta seu texto como comandos regulares do Texinfo). Isso é útil especialmente para incluir arquivos gerados automaticamente em um manual Texinfo.

Geralmente, a saída será exatamente a mesma que a entrada. Nenhuma substituição de caracteres é feita, por exemplo, todas as linhas em branco e espaços são significantes, incluindo "tabs". No manual impresso, o texto é produzido em uma fonte de largura fixa, e não recuado ou preenchido.

Escreva um comando `@verbatim` sozinho no início de uma linha. Essa linha desaparecerá da saída. Marque o final do bloco "verbatim" com um comando `@end verbatim`, também escrito sozinho no início de uma linha. O `@end verbatim` também desaparecerá da saída.

Por exemplo:

```
@verbatim
{
TAB@command com caracteres estranhos: @'e
expanda-TABme
}
@end verbatim
```

Isso produz:

```
{
        @command com caracteres estranhos: @'e
expanda-        me
}
```

Dado que as linhas contendo `@verbatim` e `@end verbatim` não produzem saída, tipicamente você deveria colocar uma linha em branco antes do `@verbatim` e outra linha em branco após o `@end verbatim`. As linhas em branco entre o início do `@verbatim` e o final do `@end verbatim` aparecerão na saída.

Você pode obter um literal "pequeno" colocando o `@verbatim` em um ambiente `@smallformat`, conforme mostrado aqui:

```
@smallformat
@verbatim
... ainda literal, mas em uma fonte menor ...
@end verbatim
```

```
@end smallformat
```

Finalmente, uma palavra de alerta: não é confiável usar `@verbatim` dentro de outras construções do Texinfo.

Veja também Seção 18.5 [`@verbatiminclude`], Página 148.

8.6 `@lisp`: Marcando um Exemplo da Lisp

O comando `@lisp` é usado para código da Lisp. Ele é um sinônimo do comando `@example`.

```
Este é um exemplo de texto escrito entre um comando @lisp e um
comando @end lisp.
```

Use `@lisp` em vez de `@example` para preservar informação relativa à natureza do exemplo. Isso é útil, por exemplo, se você escrever uma função que avalia apenas e todo o código Lisp em um arquivo do Texinfo. Então você pode usar o arquivo Texinfo como uma biblioteca Lisp. Marque o final do `@lisp` com `@end lisp` sozinho em uma linha.

8.7 `@display`: Exemplos Usando a Fonte de Texto

O comando `@display` inicia outro tipo de ambiente, onde a fonte é deixada inalterada, não trocada para máquina de escrever como com `@example`. Cada linha da entrada ainda produz uma linha de saída, e a saída ainda é recuada.

Este é um exemplo de texto escrito entre um comando `@display` e
um comando `@end display`. O comando `@display` recua o
texto, mas não o preenche.

Texinfo também fornece o ambiente `@smalldisplay`, que é como `@display`, mas usa um tamanho de fonte menor. Veja Seção 8.15 [`@small...`], Página 73.

8.8 `@format`: Exemplos Usando a Largura Total da Linha

O comando `@format` é similar a `@display`, exceto que deixa o texto sem recuo. Como `@display`, não seleciona a fonte de largura fixa.

Este é um exemplo de texto escrito entre um comando `@format` e um
comando `@end format`. Como você pode ver a partir deste exemplo,
o comando `@format` não preenche o texto.

Texinfo também fornece o ambiente `@smallformat`, que é como `@format`, mas usa um tamanho de fonte menor. Veja Seção 8.15 [`@small...`], Página 73.

8.9 `@exdent`: Desfazendo o Recuo de Uma Linha

O comando `@exdent` remove qualquer recuo que uma linha possa ter. O comando é escrito no início de uma linha e se aplica somente ao texto que segue o comando que está na mesma linha. Não use chaves em volta do texto. Em um manual impresso, o texto em uma linha `@exdent` é impresso na fonte romana.

`@exdent` normalmente é usado dentro de exemplos. Assim,

```
@example
Esta linha segue um comando @@example.
@exdent Esta linha está sem recuo.
Esta linha segue a linha sem recuo.
O @@end example vem na próxima linha.
@end example
```

produz:

```
Esta linha segue um comando @example.
```
Esta linha está sem recuo.
```
Esta linha segue a linha sem recuo.
O @end example vem na próxima linha.
```

Na prática, o comando `@exdent` raramente é usado. Geralmente, você retira o recuo do texto finalizando o exemplo e retornando a página para a sua largura normal.

`@exdent` não tem efeito em saída HTML.

8.10 `@flushleft` e `@flushright`

Os comandos `@flushleft` e `@flushright` alinham as extremidades das linhas nas margens esquerda e direita de uma página, mas não preenchem o texto. Os comandos são escritos sozinhos em linhas próprias, sem chaves. Os comandos `@flushleft` e `@flushright` são finalizados pelos comandos `@end flushleft` e `@end flushright` sozinhos em suas próprias linhas.

Por exemplo,

```
@flushleft
Este texto está
escrito flushleft.
@end flushleft
```

produz:

Este texto está
escrito flushleft.

`@flushright` produz o tipo de recuo frequentemente usado no endereço de retorno de cartas. Por exemplo,

```
@flushright
Aqui está um exemplo de texto escrito
flushright.  O comando @code{@flushright}
justifica a direita de cada linha, mas deixa a
extremidade esquerda irregular.
@end flushright
```

produz:

Aqui está um exemplo de texto escrito
flushright. O comando `@flushright`
justifica a direita de cada linha, mas deixa a
extremidade esquerda irregular.

8.11 `@raggedright`: Texto Irregular a Direita

O `@raggedright` preenche texto como de costume, mas o texto só é justificado a esquerda; a margem direita é irregular. O comando é escrito sozinho em uma linha, sem chaves. O comando `@raggedright` é finalizado por `@end raggedright` sozinho em uma linha. Esse comando não tem efeito em saída Info e HTML, onde o texto sempre é definido como irregular a direita.

O comando `@raggedright` pode ser útil com parágrafos contendo listas de comandos com nomes longos, quando se sabe antecipadamente que justificar o texto em ambas as margens fará com que o parágrafo tenha aparência ruim.

Um exemplo (originário de outro lugar neste manual):

```
@raggedright
Comandos para aspas de ângulo duplo e único:
@code{@@guillemetleft@{@}}, @code{@@guillemetright@{@}},
@code{@@guillemotleft@{@}}, @code{@@guillemotright@{@}},
@code{@@guilsinglleft@{@}}, @code{@@guilsinglright@{@}}.
@end raggedright
```

produz:

Comandos para aspas de ângulo duplo e único: `@guillemetleft{}`, `@guillemetright{}`, `@guillemotleft{}`, `@guillemotright{}`, `@guilsinglleft{}`, `@guilsinglright{}`.

8.12 `@noindent`: Omitindo Recuo

Um exemplo ou outra inclusão pode quebrar um parágrafo em segmentos. Normalmente, os formatadores recuam o texto que segue um exemplo como um parágrafo novo. Você pode evitar isso caso a caso escrevendo `@noindent` no início de uma linha, precedendo o texto de continuação. Você também pode desativar o recuo de todos os parágrafos globalmente com `@paragraphindent` (veja Seção 3.7.4 [`@paragraphindent`], Página 27).

Aqui está um exemplo mostrando como eliminar o recuo normal do texto após um `@example`, uma situação comum:

```
@example
Este é um exemplo
@end example

@noindent
Esta linha não é recuada. Como você pode ver, o
início da linha é nivelado totalmente com a
linha seguinte.
```

produz:

```
Este é um exemplo
```

Esta linha não é recuada. Como você pode ver, o
início da linha é nivelado totalmente com a
linha seguinte.

O uso padrão do `@indent` é exatamente como acima: no início do que seria de outro modo um parágrafo, para eliminar o recuo que normalmente acontece ali. O comando ou pode ser seguido por texto ou estar sozinho em uma linha. Não há razão para usá-lo em outros contextos, como no meio de um parágrafo ou dentro de um ambiente (veja Capítulo 8 [Citações e Exemplos], Página 66).

Você pode controlar o número de linhas em branco na saída do arquivo Info ajustando a entrada como desejado: uma linha contendo apenas `@noindent` não gera uma linha em branco, e também não o faz uma linha `@end` para um ambiente.

Não coloque chaves após um comando `@noindent`; elas não usadas, dado que `@noindent` é um comando usado do lado de fora de parágrafos (veja Seção A.1 [Sintaxe de Comando], Página 204).

8.13 `@indent`: Forçando o Recuo

Para complementar o comando `@noindent` (veja a seção anterior), Texinfo fornece o comando `@indent` para forçar que um parágrafo seja recuado. Por exemplo, este parágrafo (o primeiro nesta seção) está recuado usando um comando `@indent`.

E, de fato, o primeiro parágrafo de uma seção é o lugar mais provável para se usar `@indent`, para anular o comportamento normal de não recuo ali (veja Seção 3.7.4 [`@paragraphindent`], Página 27). O comando ou pode ser seguido por texto ou estar sozinho em uma linha.

Como um caso especial, quando `@indent` é usado em um ambiente onde o texto não é preenchido, ele produz um espaço de recuo de parágrafo na saída TEX. (Esses ambientes são onde uma linha de entrada produz uma linha de saída, como `@example` e `@display`; para um resumo de todos os ambientes, veja Seção 8.1 [Comandos de Cercamento de Blocos], Página 66).

Não coloque chaves após um comando `@indent`; elas não são usadas, dado que `@indent` é um comando usado do lado de fora de parágrafos (veja Seção A.1 [Sintaxe de Comando], Página 204).

8.14 `@cartouche`: Retângulos Arredondados

Em um manual impresso, o comando `@cartouche` desenha uma caixa com cantos arredondados em torno do seu conteúdo. Em HTML, um retângulo normal é desenhado. `@cartouche` não tem efeito em saída Info.

Você pode usar esse comando para realçar ainda mais um exemplo ou citação. Por exemplo, você poderia escrever um manual no qual um tipo de exemplo esteja envolvido por um "cartouche" para ênfase.

Por exemplo,

```
@cartouche
@example
% pwd
/usr/local/share/emacs
@end example
@end cartouche
```

cerca o exemplo de duas linhas com uma caixa com cantos arredondados, no manual impresso.

A saída oriunda do exemplo se parece com isto (se você estiver lendo isto em Info, você verá que o `@cartouche` não teve efeito):

```
% pwd
/usr/local/info
```

`@cartouche` também implica `@group` (veja Seção 13.9 [`@group`], Página 112).

8.15 `@small...` Comandos de Bloco

Adicionalmente ao `@example` regular e comandos similares, Texinfo tem comandos estilo de exemplo "pequeno". Esses são `@smallquotation`, `@smallindentedblock`, `@smalldisplay`, `@smallexample`, `@smallformat`, e `@smalllisp`.

Em saída Info, os comandos `@small...` são equivalentes aos seus comandos complementares não "pequeno".

Em TEX, entretanto, os comandos `@small...` produzem texto em uma fonte menor que os comandos exemplo não-small. Assim, por exemplo, exemplos de código podem conter linhas mais longas e ainda caberem em uma página sem a necessidade de serem reescritos.

Um tamanho de fonte menor também é solicitado em saída HTML, e (como de costume) retido na transliteração XML do Texinfo .

Marque o fim de um bloco `@small...` com um correspondente `@end small...` . Por exemplo, emparelhe `@smallexample` com `@end smallexample`.

Aqui está um exemplo da fonte usada pelo comando `@smallexample` (em Info, a saída será a mesma como de costume):

```
... para ter certeza de que você tem a liberdade de
distribuir cópias de software livre (e cobrar por
esse serviço se você desejar), que você receba o código
fonte ou consiga obtê-lo se quiser, que você pode
modificar o software ou usar partes dele em novos
programas livres; e que você sabe que você pode fazer
essas coisas.
```

Os comandos `@small...` usam o mesmo estilo de fonte que suas contrapartes normais: `@smallexample` e `@smalllisp` usam uma fonte de largura fixa, e os demais usam a fonte regular. Eles também tem o mesmo comportamento em outros aspectos—quando o preenchimento é feito e quando as margens são reduzidas.

Como regra geral, um documento impresso terá uma melhor aparência se você usar somente um de (por exemplo) `@example` ou `@smallexample` consistentemente dentro de um capítulo.

9 Listas e Tabelas

Texinfo tem várias maneiras de construir listas e tabelas. As listas podem ser marcadas ou numeradas; tabelas de duas colunas podem realçar os itens na primeira coluna; tabelas multi colunas também são suportadas.

9.1 Listas de Introdução

Texinfo automaticamente recua o texto em listas ou tabelas, e numera uma lista numerada. Esse último recurso é útil se você modificar a lista, dado que não precisa renumerá-la você mesmo.

As listas numeradas e tabelas se iniciam com o comando @ apropriado no início de uma linha, e finalizam-se com o correspondente `@end` sozinho em uma linha. Os comandos de tabela e lista de itens também exigem que você escreva informação de formatação na mesma linha do comando @ de inicialização.

Inicie uma lista numerada, por exemplo, com um comando `@enumerate` e finalize a lista com um comando `@end enumerate`. Inicie uma lista de itens com um comando `@itemize`, seguido na mesma linha por um comando de formatação como `@bullet`, e finalize a lista com um comando `@end itemize`.

Preceda cada elemento de uma lista com um comando `@item` ou `@itemx`.

Aqui está uma lista de itens dos diferentes tipos de tabela e listas:

- Listas de itens com e sem marcações.
- Listas numeradas, usando números ou letras.
- Tabelas de duas colunas com realçamento.

Aqui está uma lista numerada com os mesmos itens:

1. Listas de itens com e sem marcações.
2. Listas numeradas, usando números ou letras.
3. Tabelas de duas colunas com realçamento.

E aqui está uma tabela de duas colunas com os mesmos itens e seus comandos @:

`@itemize` Listas de itens com e sem marcações.

`@enumerate`
: Listas numeradas, usando números ou letras.

`@table`
`@ftable`
`@vtable` Tabelas de duas colunas, opcionalmente com indexação.

9.2 `@itemize`: Construindo Uma Lista de Itens

O comando `@itemize` produz uma sequência de "itens", cada um iniciando com uma bola ou outra marca dentro da margem esquerda, e geralmente recuados.

Inicie uma lista de itens escrevendo `@itemize` no início de uma linha. Siga o comando, na mesma linha, com um caractere ou um comando do Texinfo que gera uma marca. Geralmente, você usará `@bullet` após `@itemize`, mas você pode usar `@minus`, ou qualquer comando ou caractere que resulte em um caractere único no arquivo Info. (Quando você escrever o comando de marca como `@bullet` após um comando `@itemize`, você pode omitir o '`{}`'). Se você não

especificar um comando de marca, o padrão é `@bullet`. Se você não quer nenhuma marca, mas ainda quer itens lógicos, use `@w{}` (nesse caso as chaves são obrigatórias).

Após o `@itemize`, escreva seus itens, cada um iniciando com `@item`. O texto pode seguir na mesma linha que `@item`. O texto de um item pode continuar por mais que um parágrafo.

Deveria existir ao menos um `@item` dentro do ambiente `@itemize`. Se nenhum estiver presente, `makeinfo` emite um alerta. Se você apenas deseja texto recuado e não uma lista de itens, use `@indentedblock`; veja Seção 8.3 [@indentedblock], Página 68.

As entradas de índice e comentários que são dados antes de um `@item` incluindo o primeiro, são automaticamente movidos (internamente) para após o `@item`, de forma que a saída esteja como esperada. Historicamente essa tem sido uma prática comum.

Geralmente, você deveria colocar uma linha em branco entre os itens. Isso coloca uma linha em branco no arquivo Info. (O TEX insere o espaço vertical apropriado em qualquer caso). Exceto quando as entradas são muito breves, esses espaços em branco fazem a lista ter uma aparência melhor.

Aqui está um exemplo do uso de `@itemize`, seguido pela saída que esse comando produz. O `@bullet` produz um '`*`' em Info e um ponto redondo em outros formatos de saída.

```
@itemize @bullet
@item
Algum texto para foo.

@item
Algum texto
para bar.
@end itemize
```

Isso produz:

- Algum texto para foo.
- Algum texto para bar.

As listas de itens podem ser incorporadas em outras listas de itens. Aqui está uma lista marcada com traços embutida em uma lista marcada com marcadores:

```
@itemize @bullet
@item
Primeiro item.

@itemize @minus
@item
Item interno.

@item
Segundo item interno.
@end itemize

@item
Segundo item externo.
@end itemize
```

Isso produz:

- Primeiro item.
  - Item interno.
  - Segundo item interno.
- Segundo item externo.

9.3 `@enumerate`: Fazendo Uma Lista Numerada ou Uma Com Letras

`@enumerate` é como `@itemize` (veja Seção 9.2 [`@itemize`], Página 75), exceto que os rótulos nos itens são números inteiros sucessivos ou letras em vez de marcadores.

Escreva o comando `@enumerate` no início de uma linha. O comando não exige um argumento, mas aceita ou um número ou uma letra como uma opção. Sem um argumento, `@enumerate` inicia a lista com o número '`1`'. Com um argumento numérico, como '`3`', o comando inicia a lista com aquele número. Com uma letra maiúscula ou minúscula, como '`A`' ou '`a`', o comando inicia a lista com aquela letra.

Escreva o texto da lista numerada da mesma maneira que uma lista de itens: escreva a linha iniciando com `@item` no início de cada item na enumeração. Está OK ter texto seguindo o `@item`, e o texto para um item pode continuar por vários parágrafos.

Você deveria colocar uma linha em branco entre as entradas na lista. Isso geralmente torna mais fácil ler o arquivo Info.

Aqui está um exemplo do `@enumerate` sem um argumento:

```
@enumerate
@item
Causas subjacentes.

@item
Causas imediatas.
@end enumerate
```

Isso produz:

1. Causas subjacentes.
2. Causas imediatas.

Aqui está um exemplo com um argumento de *3*:

```
@enumerate 3
@item
Causas predisponentes.

@item
Causas precipitantes.

@item
Causas perpetuantes.
@end enumerate
```

Isso produz:

3. Causas predisponentes.
4. Causas precipitantes.
5. Causas perpetuantes.

Aqui está um resumo breve das alternativas. O resumo é construído usando `@enumerate` com um argumento de `a`.

a. `@enumerate`

   Sem um argumento, produz uma lista numerada, com o primeiro item numerado 1.

b. `@enumerate` *`unsigned-integer`*

   Com um argumento numérico (não sinalado), inicia uma lista numerada com aquele número. Você pode usar isso para continuar uma lista que você interrompeu com outro texto.

c. `@enumerate` *`upper-case-letter`*

   Com uma letra maiúscula como argumento, inicia uma lista na qual cada item é marcado por uma letra, iniciando com aquela letra maiúscula.

d. `@enumerate` *`lower-case-letter`*

   Com uma letra minúscula como argumento, inicia uma lista na qual cada item é marcado por uma letra, iniciando com aquela letra minúscula.

Você também pode aninhar listas enumeradas, como em uma estrutura de tópicos.

9.4 Fazendo Uma Tabela de Duas Colunas

`@table` é similar a `@itemize` (veja Seção 9.2 [`@itemize`], Página 75), mas permite especificar um nome ou uma linha de título para cada item. O comando `@table` é usado para produzir tabelas de duas colunas, e é especialmente útil para glossários, exibições explicativas, e resumos de opções de linha de comando.

9.4.1 Usando o Comando `@table`

Use o comando `@table` para produzir uma tabela de duas colunas. Esse comando tipicamente é usado quando você tem uma lista de itens e um texto breve com cada um, como uma lista de definições.

Escreva o comando `@table` no início de uma linha, após uma linha em branco, e siga-o, na mesma linha, com um argumento que é um comando de "indicação", como `@code`, `@samp`, `@var`, `@option`, ou `@kbd` (veja Seção 7.1 [Indicando], Página 56). Esse comando será aplicado ao texto na primeira coluna. Por exemplo, `@table @code` fará com que o texto na primeira coluna seja produzido como se ele tivesse sido o argumento para um comando `@code`.

Você pode usar o comando `@asis` como um argumento para `@table`. `@asis` é um comando que não faz nada: se você usar esse comando após `@table`, as entradas da primeira coluna são exibidas sem realce adicional ("como está").

O comando `@table` funciona com outros comandos além daqueles mencionados explicitamente aqui. Entretanto, você só pode usar comandos predefinidos do Texinfo que recebam um argumento entre chaves. Você não pode confiavelmente usar um comando novo definido com `@macro`, embora um `@alias` seja aceitável (para um comando predefinido adequado). Veja Capítulo 17 [Definindo Novos Comandos do Texinfo], Página 137.

Inicie cada entrada da tabela com um comando `@item` no início de uma linha. Escreva o texto para a primeira coluna na mesma linha que o comando `@item`. Escreva o texto para a segunda coluna na linha seguinte à linha `@item` e em linhas subsequentes. Você pode escrever quantas linhas de texto de apoio desejar, até mesmo vários parágrafos. Mas, somente o texto na mesma linha que `@item` será colocado na primeira coluna (incluindo quaisquer notas de rodapé). Você não precisa digitar nada para uma segunda coluna vazia.

Normalmente, você deveria colocar uma linha em branco antes de uma linha `@item`, (exceto a primeira). Isso coloca uma linha em branco no arquivo Info. Exceto quando as entradas são muito breves, uma linha em branco parece melhor. Finalize a tabela com uma linha consistindo de `@end table`, seguido por uma linha em branco. O TEX sempre iniciará um parágrafo novo após a tabela, de forma que a linha em branco é necessária para que a saída Info seja análoga.

Por exemplo, a tabela seguinte realça o texto na primeira coluna com o comando `@samp`:

```
@table @samp
@item foo
Este é o texto para
@samp{foo}.

@item bar
Texto para @samp{bar}.
@end table
```

Isso produz:

'`foo`' Este é o texto para '`foo`'.

'`bar`' Texto para '`bar`'.

Se você quer listas dois ou mais itens nomeados com um bloco único de texto, use o comando `@itemx`. (Veja Seção 9.4.3 [`@itemx`], Página 79).

O comando `@table` (veja Seção 9.4.1 [`@table`], Página 78) não é suportado dentro do `@display`. Dado que `@display` é orientado a linha, não faz sentido usá-los juntos. Se você deseja recuar uma tabela, tente `@quotation` (veja Seção 8.2 [`@quotation`], Página 67) ou `@indentedblock` (veja Seção 8.3 [`@indentedblock`], Página 68).

9.4.2 `@ftable` e `@vtable`

Os comandos `@ftable` e `@vtable` são os mesmos que o comando `@table`, exceto que `@ftable` automaticamente insere cada um dos itens na primeira coluna da tabela no índice de funções e `@vtable` automaticamente insere cada um dos itens na primeira coluna da tabela no índice de variáveis. Apenas os itens na mesma linha que os comandos `@item` ou `@itemx` são indexados, e são indexados exatamente na forma em que aparecem nessa linha. Veja Capítulo 11 [Índices], Página 89, para mais informação acerca de índices.

Inicie uma tabela de duas colunas usando `@ftable` ou `@vtable` escrevendo o comando @ no início de uma linha, seguido na mesma linha por um argumento que é um comando do Texinfo, como `@code`, exatamente como você faria para um comando `@table`; e termine a tabela com um comando `@end ftable` ou `@end vtable` sozinho em uma linha.

Veja o exemplo para `@table` na seção anterior.

9.4.3 `@itemx`: Segundo e Itens Subsequentes

Use o comando `@itemx` dentro de uma tabela quando você tiver duas ou mais entradas de primeira coluna para o mesmo item, cada uma das quais deveria aparecer sozinha em uma linha.

Use `@item` para a primeira entrada, e `@itemx` para todas as entradas subsequentes; `@itemx` sempre deve seguir um comando `@item`, sem linhas em branco entre eles.

O comando `@itemx` funciona exatamente como `@item`, exceto que não gera espaço vertical extra acima do texto da primeira coluna. Se você tiver múltiplos comandos `@itemx` consecutivos, não insira quaisquer linhas em branco entre eles.

Por exemplo,

```
@table @code
@item upcase
@itemx downcase
Essas duas funções aceitam um caracter ou uma sequência de caracteres
como argumento, e retornam o correspondente caracter ou a sequência de
caracteres escritos em letras maiúsculas (minúsculas).
@end table
```

Isso produz:

`upcase`
`downcase` Essas duas funções aceitam um caractere ou uma sequência de caracteres como argumento, e retornam o correspondente caractere ou a sequência de caracteres escritos em letras maiúsculas (minúsculas).

(Note também que esse exemplo ilustra texto de apoio multi linha em uma tabela de duas colunas).

9.5 `@multitable`: Tabelas Multi Colunas

`@multitable` permite que você construa tabelas com qualquer número de colunas, com cada coluna tendo qualquer largura que você goste.

Você define as larguras da coluna em uma linha `@multitable` sozinha, e escreve cada linha da atual tabela seguindo um comando `@item`, com colunas separadas por um comando `@tab`. Finalmente, `@end multitable` completa a tabela. Detalhes nas seções abaixo.

9.5.1 Larguras de Colunas Multi Tabelas

Você pode definir as larguras de coluna para uma multi tabela de duas maneiras: como frações do comprimento da linha; ou com uma linha de protótipo. A mesclagem dos dois métodos não é suportada. Em qualquer caso, as larguras são definidas inteiramente na mesma linha que o comando `@multitable`.

1. Para especificar larguras de coluna como frações do comprimento de linha, escreva `@columnfractions` e os números decimais (presumivelmente menor que 1; um zero no início é permitido e ignorado) após o comando `@multitable`, como em:

   ```
   @multitable @columnfractions .33 .33 .33
   ```

   As frações não precisam somar exatamente a 1.0, como essas não fazem. Isso permite produzir tabelas que não precisam do comprimento total da linha.

2. Para especificar uma linha de protótipo, escreva a entrada mais longa para cada coluna entre chaves após o comando `@multitable`. Por exemplo:

   ```
   @multitable {algum texto para a coluna um} {para a coluna dois}
   ```

   A primeira coluna terá então a largura do conjunto de caracteres 'algum texto para a coluna um' e a segunda coluna a largura de 'para a coluna dois'.

   As entradas de protótipo não precisam aparecer na própria tabela.

   Embora tenhamos usado texto simples nesse exemplo, as entradas de protótipo podem conter comandos do Texinfo; comandos de marcação, como `@code`, são particularmente prováveis de serem úteis.

9.5.2 Linhas de Multi Tabelas

Após o comando `@multitable` definindo as larguras de coluna (veja a seção anterior), você inicia cada linha no corpo de uma multi tabela com `@item`, e separa as entradas de coluna com `@tab`. As quebras de linha não são especiais dentro do corpo da tabela, e você pode quebrar linhas de entrada em seu arquivo fonte se necessário.

Você também pode usar `@headitem` em vez de `@item` para produzir uma *linha de cabeçalho*. A saída do TEX para tal linha é em negrito, e a saída HTML e Docbook usa a marcação `<thead>`. Em Info, a linha de cabeçalho é seguida por uma linha separadora feita de traços (caracteres '`-`').

O comando `@headitemfont` pode ser usado em modelos quando as entradas em uma linha `@headitem` precisarem ser usadas em um modelo. É um sinônimo para `@b`, mas usar

`@headitemfont` evita qualquer dependência daquele estilo específico de fonte, no caso de fornecemos uma maneira de alterá-lo no futuro.

Aqui está um exemplo completo de uma tabela multi coluna (o texto é originário de *O Manual do GNU Emacs*, veja Seção "Splitting Windows" em *O Manual do GNU Emacs*):

```
@multitable @columnfractions .15 .45 .4
@headitem Tecla @tab Comando @tab Descrição
@item C-x 2
@tab @code{split-window-vertically}
@tab Divide a janela selecionada em duas janelas,
com uma acima da outra.
@item C-x 3
@tab @code{split-window-horizontally}
@tab Divide a janela selecionada em duas janelas
posicionadas lado a lado.
@item C-Mouse-2
@tab
@tab Na linha de modo ou barra de rolagem de uma janela,
divide aquela janela.
@end multitable
```

produz:

Tecla	**Comando**	**Descrição**
C-x 2	`split-window-vertically`	Divide a janela selecionada em duas janelas, com uma acima da outra.
C-x 3	`split-window-horizontally`	Divide a janela selecionada em duas janelas posicionadas lado a lado.
C-Mouse-2		Na linha de modo ou barra de rolagem de uma janela, divide aquela janela.

10 Exibições Especiais

Os comandos neste capítulo permitem escrever texto que é especialmente exibido (permissão de formato de saída), fora do fluxo normal de documento.

Um conjunto de tais comandos é para a criação de "flutuadores", isto é, figuras, tabelas, e afins, configuradas a partir do texto principal, possivelmente numeradas, rotuladas, e (ou) referenciadas a partir de outro lugar no documento. As imagens frequentemente são incluídas nessas exibições.

Outro grupo de comandos é para a criação de notas de rodapé em Texinfo.

10.1 Flutuações

Um *flutuador* é uma exibição que é configurada a partir do texto principal. É tipicamente rotulado como sendo uma "Figura", "Tabela", "Exemplo" ou algum tipo similar.

Um flutuador é assim nomeado porque, teoricamente, pode ser movido para o pé ou topo da página atual, ou para uma página seguinte, na saída impressa. (Flutuação não faz sentido em outros formatos de saída). Na presente versão de Texinfo, entretanto, essa flutuação infelizmente ainda não está implementada. Em vez disso, o material flutuante é simplesmente produzido na localização atual, mais ou menos como se fosse um `@group` (veja Seção 13.9 [`@group`], Página 112).

10.1.1 `@float` [*tipo*][,*rótulo*]: Material Flutuante

Para produzir material flutuante, coloque o material que você quer que seja exibido separado entre os comandos `@float` e `@end float`, sozinhos em suas linhas.

Material flutuante geralmente usa `@image` para exibir um gráfico já existente (veja Seção 10.2 [Imagens], Página 84), ou `@multitable` para exibir uma tabela (veja Seção 9.5 [Tabelas Multi Colunas], Página 80). Entretanto, o conteúdo do flutuador podem ser qualquer coisa. Aqui está um exemplo com texto simples:

```
@float Figura,fig:ex1
Este é um flutuador de exemplo.
@end float
```

E a saída:

Este é um flutuador de exemplo.

Figura 10.1

Como mostrado no exemplo, `@float` recebe dois argumentos (separados por vírgula), *tipo* e *rótulo*. Ambos são opcionais.

tipo Especifica a classe do flutuador; tipicamente uma palavra como "Figura", "Tabela", etc. Se não for dada, e *rótulo* for, quaisquer referências cruzadas simplesmente usarão um número despojado.

rótulo Especifica um rótulo de referência cruzada para este flutuador. Se dado, a este flutuador é automaticamente dado um número, e aparecerá em qualquer saída `@listoffloats` (veja Seção 10.1.3 [`@listoffloats`], Página 83). Referências cruzadas a *rótulo* são permitidas.

Por outro lado, se *rótulo* não for dado, então o flutuador não será numerado e consequentemente não aparecerá na saída `@listoffloats` ou ser referenciável.

Ordinariamente, você especifica ambos *tipo* e *rótulo*, para obter um flutuador rotulado e numerado.

Em Texinfo, todos os flutuadores são numerados na mesma maneira: com o número do capítulo (ou letra do apêndice), um ponto, e o número do flutuador, o qual simplesmente conta 1, 2, 3, ..., e é zerado a cada capítulo. Cada tipo de flutuador é contado independentemente.

Os flutuadores dentro de `@unnumbered`, ou fora de qualquer capítulo, são numerados simplesmente consecutivamente a partir de 1.

Essas convenções de numeração não são, atualmente, modificáveis.

10.1.2 `@caption` e `@shortcaption`

Você pode escrever um `@caption` em qualquer lugar dentro de um ambiente `@float`, para definir uma legenda para o flutuador. Não é permitido em qualquer outro contexto. `@caption` recebe somente um argumento, envolvido em chaves. Aqui está um exemplo:

```
@float
Um exemplo de flutuador, com legenda.
@caption{Legenda para exemplo de flutuador.}
@end float
```

A saída é:

Um exemplo de flutuador, com legenda.

Legenda para exemplo de flutuador.

`@caption` pode aparecer em qualquer lugar dentro do flutuador; `@caption` não é processado até o `@end float`. O texto da legenda geralmente é uma frase ou duas, porém pode consistir de vários parágrafos se necessário.

Na saída, a legenda sempre aparece abaixo do flutuador; isso atualmente não é modificável. A legenda é precedida pelo número e (ou) tipo do flutuador, conforme especificado para o comando `@float` (veja-se a seção anterior).

O comando `@shortcaption` igualmente pode ser usado somente dentro de `@float`, e recebe somente um argumento dentro de chaves. O texto curto da legenda usado em vez do texto da legenda em uma lista de flutuadores (veja-se a próxima seção). Assim, você pode escrever uma legenda longa para o documento principal, e um título curto aparecer na lista de flutuadores. Por exemplo:

```
@float
... conforme acima ...
@shortcaption{Texto para a lista de flutuadores.}
@end float
```

O texto para `@shortcaption` pode não conter comentários (`@c`), texto literal (`@verb`), ambientes como `@example`, notas de rodapé (`@footnote`) ou outras construções complexas. As mesmas restrições se aplicam ao `@caption` a menos que exista um `@shortcaption`.

10.1.3 `@listoffloats`: Tabelas de Conteúdos para Flutuadores

Você pode escrever um comando `@listoffloats` para gerar uma lista de flutuadores para um dado tipo de flutuador (veja Seção 10.1.1 [`@float`], Página 82), semelhante ao Sumário geral do documento. Tipicamente, o comando é escrito no nó `@unnumbered` dele próprio para prover um cabeçalho e estrutura, mais como `@printindex` (veja Seção 11.4 [Imprimindo Índices e Menus], Página 91).

`@listoffloats` recebe um argumento opcional, o tipo de flutuador. Eis um exemplo:

```
@node List of Figures
@unnumbered List of Figures
@listoffloats Figure
```

E aqui está com o que se parece a saída a partir de `@listoffloats`, dada a figura de exemplo anterior neste capítulo (a saída Info está formatada como um menu):

Sem qualquer argumento, `@listoffloats` gera uma lista de flutuadores para os quais nenhum tipo de flutuador foi especificado, isto é, sem primeiro argumento para o comando `@float` (veja Seção 10.1.1 [`@float`], Página 82).

Cada linha na lista de flutuadores contém o tipo de flutuador (se algum), o número do flutuador, e a legenda, se alguma—o argumento `@shortcaption`, se foi especificado, do contrário o argumento `@caption`. Em Info, o resultado é um menu onde cada flutuador pode ser selecionado. Em HTML, cada linha é um link para o flutuador. Em saída impressa, o número da página é incluído.

Flutuadores não numerados (aqueles sem rótulos de referência cruzada) são omitidos da lista de flutuadores.

10.2 Inserindo Imagens

Você pode inserir uma imagem dada em um arquivo externo com o comando `@image`. Apesar de imagens poderem ser usadas em qualquer lugar, incluindo o meio de um parágrafo, nós as descrevemos neste capítulo dado que elas (as imagens) são mais frequentemente parte de uma figura exibida ou exemplo.

10.2.1 Sintaxe da Imagem

Aqui está a sinopse do comando `@image`:

```
@image{filename[, width[, height[, alttext[, extension]]]]}
```

O argumento *filename* é obrigatório, e não deve ter uma extensão, pois os diferentes processadores suportam formatos diferentes:

- TEX (saída DVI) lê o arquivo *filename*`.eps` (formato PostScript Encapsulado).
- pdfTEX Lê *filename*`.pdf`, *filename*`.png`, *filename*`.jpg`, ou *filename*`.jpeg` (nessa ordem). Também tenta as versões em letras maiúsculas das extensões. O formato PDF não suporta imagens EPS, de forma que essas devem ser convertidas primeiro.
- Para Info, `makeinfo` inclui *filename*`.txt` literal (mais ou menos como se estivesse em `@verbatim`). A saída Info também pode incluir uma referência para *filename*`.png` ou *filename*`.jpg`. (Veja-se abaixo).
- Para HTML, `makeinfo` emite uma referência para *filename*`.png`, *filename*`.jpg`, *filename*`.jpeg` ou *filename*`.gif` (nessa ordem). Se nenhum desses existir, então dá um erro, e emite uma referência para *filename*`.jpg` de qualquer maneira.
- Para Docbook, `makeinfo` emite referências para *filename*`.eps`, *filename*`.gif` *filename*`.jpeg`, *filename*`.jpg`, *filename*`.pdf`, *filename*`.png` e *filename*`.svg`, para cada arquivo encontrado. Também, *filename*`.txt` é incluído literalmente, se presente. (Supõe-se que o processador Docbook subsequente escolha o adequado).
- Para saída Info e HTML, `makeinfo` usa o quinto argumento opcional *extension* para `@image` para a extensão de nome de arquivo, se for especificado e o arquivo for encontrado. Qualquer ponto inicial deveria ser incluído em *extension*. Por exemplo:

```
@image{foo,,,,.xpm}
```

Se você deseja instalar arquivos de imagens para uso por leitores Info, então recomenda-se colocá-los em um subdiretório como '*foo*`-figures`' para um pacote *foo*. A cópia dos arquivos para `$(infodir)/`*foo*`-figures/` deveria ser feita em seu `Makefile`.

Os argumentos *width* e *height* estão descritos na próxima seção.

Para saída TEX, se uma imagem for a única coisa em um parágrafo, ela ordinariamente será exibida em uma linha própria, respeitando o recuo atual do ambiente, porém sem o recuo normal do parágrafo. Se você quiser a imagem centralizada, então use `@center` (veja Seção 3.4.2 [`@titlefont @center @sp`], Página 19).

Para saída HTML, `makeinfo` configura o *atributo alt* para imagens inline para o (quarto) argumento opcional *alttext* para `@image`, se fornecido. Se não fornecido, então `makeinfo` usa o nome completo de arquivo da imagem sendo exibida. A *alttext* é processada como texto Texinfo, de forma que caracteres especiais, tais como '`"`' e '`<`' e '`&`', são escapados na saída HTML; também, você pode obter uma sequência `alt` vazia com `@-` (um comando que não produz saída; veja Seção 13.3 [`@- @hyphenation`], Página 111).

Para saída Info, a sequência `alt` também é processada como texto de Texinfo e saída. Nesse caso, '`\`' é escapada como '`\\`' e '`"`' como '`\"`'; outros escapes não são feitos.

Em saída Info, `makeinfo` escreve uma referência para o arquivo binário de imagem (tentando *filename* com a extensão *`extension`*, *`.extension`*, `.png`, ou `.jpg`, nessa ordem) se um existir. Também incluir literalmente o arquivo `.txt` se um existir. Dessa maneira, os leitores Info que podem exibir imagens (tais como o navegador Emacs Info, sendo executado sob X) podem fazer isso, ao passo que leitores Info que somente podem usar texto (tais como o leitor autônomo Info) podem exibir a versão textual.

A implementação para isso é colocar o seguinte construtor na saída de Info:

```
^@^H[image src="binaryfile" text="txtfile"
            alt="alttext ... ^@^H]
```

onde '`^@`' e '`^H`' significam os caracteres atuais de controle "null" e "backspace". Se um dos arquivos não estiver presente, então o correspondente argumento é omitido.

A justificativa para se mencionar isso aqui é que navegadores Info mais antigos (essa característica foi introduzida na versão 4.6 de Texinfo) exibirão o acima literalmente, o que, apesar de não ser agradável, não deveria ser danoso.

10.2.2 Escalonamento da Imagem

Os argumentos opcionais *width* e *height* para o comando `@image` (veja-se a seção anterior) especificam o tamanho para o qual escalonar a imagem. Eles somente são levados em consideração no TEX. Se nenhum for especificado, a imagem será apresentada no tamanho natural dela (fornecido no arquivo); se somente um for especificado, o outro será escalonado proporcionalmente; e se ambos forem especificados, ambos serão respeitados, portanto provavelmente distorcendo a imagem original mudando a taxa de aspecto dela.

width e *height* podem ser especificados usando qualquer dimensão TEX válida, a saber:

pt	ponto (72.27pt = 1in)
pc	pica (1pc = 12pt)
bp	ponto grande (72bp = 1in)
in	polegada
cm	centímetro (2.54cm = 1in)
mm	milímetro (10mm = 1cm)
dd	ponto de didôt (1157dd = 1238pt)
cc	cicero (1cc = 12dd)
sp	ponto escalonado (65536sp = 1pt)

Por exemplo, o seguinte dimensionará um arquivo `ridt.eps` para uma polegada verticalmente, com a largura dimensionada proporcionalmente:

```
@image{ridt,,1in}
```

Para que o `@image` funcione com o TEX, o arquivo `epsf.tex` deve ser instalado em algum lugar onde o TEX possa encontrá-lo. (O local padrão é *texmf*`/tex/generic/dvips/epsf.tex`, onde *texmf* é a raiz da árvore de diretórios do teu TEX). Esse arquivo está incluído na distribuição Texinfo e também está disponível a partir de `ftp://tug.org/tex/epsf.tex`, entre outros lugares.

`@image` pode ser usado dentro de uma linha e também para figuras exibidas. Portanto, se você pretende que ele seja exibido, assegure-se de deixar uma linha em branco antes do comando, ou a saída gerada será executada no texto anterior.

Atualmente, o dimensionamento de imagem está implementado somente no TEX, não em HTML ou qualquer outro tipo de saída gerada.

10.3 Notas de Rodapé

Uma *nota de rodapé* serve para uma referência que documenta ou elucida o texto principal.[1].

As notas de rodapé distraem; use-as com moderação, no máximo, e é melhor evitá-las completamente. As referências bibliográficas padrão geralmente são melhor colocadas em uma bibliografia, ao final de um documento, em vez de nas notas de rodapé.

10.3.1 Comandos de Notas de Rodapé

No Texinfo, as notas de rodapé são criadas com o comando `@footnote`. Esse comando é seguido imediatamente por uma chave esquerda, depois pelo texto da nota de rodapé e por uma chave direita finalizante. As notas de rodapé podem ter qualquer comprimento (elas serão divididas entre páginas, se necessário), mas geralmente são curtas. O modelo é:

```
texto principal@footnote{texto da nota de rodapé}
```

Conforme mostrado aqui, o comando `@footnote` deveria vir logo após o texto sendo anotado, sem espaço intermediário; caso contrário, o marcador de nota de rodapé poderá acabar iniciando uma linha.

Por exemplo, esta cláusula é seguida por uma amostra de nota de rodapé[2]; no fonte do Texinfo, se parece com isto:

```
...uma amostra de nota de rodapé@footnote{Aqui está a amostra de
nota de rodapé.}; no fonte do Texinfo...
```

Como você pode ver, esse fonte inclui dois sinais de pontuação próximos um do outro; nesse caso, '`.};`' é a sequência. Isso é normal (o primeiro termina a nota de rodapé e o segundo pertence à frase sendo anotada), de forma que não se preocupe, pois pode parecer estranho. (Outro estilo, perfeitamente aceitável, é colocar a nota de rodapé depois da pontuação pertencente à frase, como em '`;@footnote{...`').

Em um manual ou livro impresso, a marca de referência para uma nota de rodapé é um número pequeno e sobrescrito; o texto da nota de rodapé aparece na parte inferior da página, abaixo de uma linha horizontal.

No Info, a marca de referência para uma nota de rodapé é um par de parênteses com o número da nota de rodapé entre eles, assim: '`(1)`'. A marca de referência é seguida por um link

[1] Uma nota de rodapé deveria complementar ou expandir o texto principal, mas o(a) leitor(a) não deveria precisar ler uma nota de rodapé para entender o texto principal. Para uma discussão completa das notas de rodapé, veja-se *The Chicago Manual of Style*, publicado pela University of Chicago Press

[2] Aqui está a amostra de nota de rodapé.

de referência cruzada para o texto da nota de rodapé se as notas de rodapé forem colocadas em nós separados (veja Seção 10.3.2 [Estilos de Notas de Rodapé], Página 87).

Na saída gerada HTML, as referências às notas de rodapé geralmente são marcadas com um número pequeno e sobrescrito que é renderizado como um link de hipertexto para o texto da nota de rodapé.

As notas de rodapé não podem ser aninhadas e não podem aparecer em títulos de seção de qualquer tipo ou em outros locais "incomuns".

Uma dica final: as notas de rodapé no argumento de um comando `@item` para um `@table` devem estar inteiramente na mesma linha que o `@item` (como de costume). Veja Seção 9.4 [Tabelas de Duas Colunas], Página 78.

10.3.2 Estilos de Notas de Rodapé

O Info tem dois estilos de nota de rodapé, que determinam onde o texto da nota de rodapé está localizado:

- No estilo de nó 'Fim', todas as notas de rodapé de um nó são colocadas no final desse nó. As notas de rodapé são separadas do resto do nó por uma linha de traços com a palavra '`Notas de rodapé`' dentro dela. Cada nota de rodapé começa com uma marca de referência '`(n)`'.

  Aqui está um exemplo da saída gerada do Info para uma nota de rodapé no estilo de fim de nó:

  ```
  --------- Notas de rodapé ---------


  (1)  Aqui está uma amostra de nota de rodapé.
  ```

- No estilo de nó 'Separado', todas as notas de rodapé para um nó são colocadas em um nó próprio construído automaticamente. Nesse estilo, uma "referência de nota de rodapé" segue cada marca de referência '`(n)`' no corpo do nó. A referência da nota de rodapé é na verdade uma referência cruzada que você usa para chegar ao nó da nota de rodapé.

  O nome do nó com as notas de rodapé é construído anexando-se '`-Footnotes`' ao nome do nó que contém as notas de rodapé. (Consequentemente, o nó das notas de rodapé para o nó `Footnotes` é `Footnotes-Footnotes`)! O nó das notas de rodapé tem um ponteiro de nó 'Acima' que leva de volta ao nó ancestral dele.

  Aqui está como a primeira nota de rodapé neste manual aparenta depois de ser formatada para o Info no estilo de nó separado:

  ```
  Arquivo: texinfo.info  Nó: Visão Geral-Footnotes, Acima: Visão Geral

  (1) A primeira sílaba de "Texinfo" é pronunciada como "speck", não
  "hex". ...
  ```

A menos que teu documento tenha notas de rodapé longas e importantes (como em, por digase, *Decline and Fall ...* de Gibbon), nós recomendamos o estilo '`end`', pois ele é mais simples para leitores(as) seguirem.

Use o comando `@footnotestyle` para especificar um estilo de nota de rodapé de um arquivo do Info. Escreva esse comando no início de uma linha seguido por um argumento, seja '`end`' para o estilo de nó final ou '`separate`' para o estilo do nó separado.

Por exemplo,

```
@footnotestyle end
```

ou

```
@footnotestyle separate
```

Escreva um comando `@footnotestyle` antes ou logo depois da linha de fim de cabeçalho no início de um arquivo Texinfo. (Você deveria incluir qualquer comando `@footnotestyle` entre

as linhas de início de cabeçalho e fim de cabeçalho, de forma que os comandos de formatação de região formatem as notas de rodapé conforme especificado).

Em HTML, quando o estilo da nota de rodapé for '`end`', ou se a saída gerada não for dividida, as notas de rodapé são colocadas no final da saída gerada. Se configurado como '`separate`' e a saída gerada for dividida, elas serão colocadas em um arquivo separado.

11 Índices

Usando o Texinfo, você consegue gerar índices sem ter que ordenar e agrupar as entradas manualmente. Em um índice, as entradas são listadas em ordem alfabética, juntamente com informações acerca de como encontrar a discussão de cada entrada. Em um manual impresso, essas informações consistem de números de páginas. Em um arquivo Info, essa informação é uma entrada de menu que leva para o primeiro nó referenciado.

O Texinfo fornece vários tipos predefinidos de índice: um índice para funções, um índice para variáveis, um índice para conceitos e assim por diante. Você pode combinar índices ou usá-los para outros fins que não os canônicos. Por último, você consegue definir teus próprios novos índices.

11.1 Índices Predefinidos

Texinfo fornece seis índices predefinidos. Aqui estão os significados nominais deles, abreviações e os correspondentes comandos de entrada de índice:

‘`cp`’ (`@cindex`) índice de conceito, para conceitos gerais.

‘`fn`’ (`@findex`) índice de função, para nomes de funções e semelhantes a funções (como pontos de entrada de bibliotecas).

‘`ky`’ (`@kindex`) índice de pressionamento de tecla, para comandos de teclado.

‘`pg`’ (`@pindex`) índice de programas, para nomes de programas.

‘`tp`’ (`@tindex`) índice de tipo de dados, para nomes de tipos (como estruturas definidas em arquivos de cabeçalho).

‘`vr`’ (`@vindex`) índice de variáveis, para nomes de variáveis (como variáveis globais de bibliotecas).

Nem todo manual precisa de tudo isso, e a maioria dos manuais usa somente dois ou três, no máximo. O presente manual, por exemplo, tem dois índices: um índice de conceito e um índice de comando @ (que na verdade é o índice de função, mas é chamado de índice de comando no cabeçalho de capítulo).

Você não é obrigado(a) a usar os índices predefinidos estritamente para os fins canônicos deles. Por exemplo, suponha que você deseje indexar algumas macros do preprocessador C. Você poderia colocá-las no índice de função junto com as funções reais, apenas escrevendo comandos `@findex` para elas; então, quando você imprimir o “Índice de Função” como um capítulo não numerado, você poderia dar a ele o título de ‘Índice de Função e Macro’ e tudo seria consistente para o(a) leitor(a).

Por outro lado, é melhor não se afastar muito do significado dos índices predefinidos. Do contrário, caso o teu texto seja combinado com outro texto proveniente de outros manuais, as entradas de índice não corresponderão. Em vez disso, defina teu próprio novo índice (veja Seção 11.6 [Novos Índices], Página 93).

Nós recomendamos ter um índice no documento final sempre que possível, independentemente de quantos índices de fonte você usar, pois assim os(as) leitores(as) terão somente um lugar para procurar. Dois ou mais índices de fonte podem ser combinados em um índice de saída gerada, usando-se os comandos `@synindex` ou `@syncodeindex` (veja Seção 11.5 [Combinando Índices], Página 92).

11.2 Definindo as Entradas de um Índice

Os dados para criar um índice vem de muitos comandos individuais de indexação espalhados pelo arquivo fonte do Texinfo. Cada comando diz para adicionar uma entrada para um índice específico; depois da formatação, o índice fornecerá o número da página atual ou o nome do nó como a referência.

Uma entrada de índice consiste de um comando de indexação no início de uma linha seguido, no restante da linha, pela entrada.

Por exemplo, esta seção começa com as seguintes cinco entradas para o índice de conceito:

```
@cindex Definindo entradas de indexação
@cindex Entradas de índice, definindo
@cindex Entradas para um índice
@cindex Especificando entradas de índice
@cindex Criando entradas de índice
```

Cada índice predefinido tem o próprio comando dele de indexação—`@cindex` para o índice de conceito, `@findex` para o índice de função e assim por diante, conforme listado na seção anterior.

As entradas do índice deveriam preceder o material visível que está sendo indexado. Por exemplo:

```
@cindex alô
Alô, você!
```

Entre outros motivos, dessa forma, seguir os links de indexação (em qualquer contexto) acaba antes do material, onde os(as) leitores(as) querem estar, e não depois.

Por padrão, as entradas para um índice de conceito são impressas em uma fonte romana pequena e as entradas para os outros índices são impressas em uma fonte `@code` pequena. Você pode mudar a forma como parte de uma entrada é impressa com os comandos usuais do Texinfo, como `@file` para nomes de arquivos (veja Capítulo 7 [Marcando Texto], Página 56) e `@r` para a fonte normal romana (veja Seção 7.2.3 [Fontes], Página 65).

Para a saída gerada impressa, você pode especificar uma chave explícita de ordenação para uma entrada de índice usando `@sortas` imediatamente depois do comando de índice. Por exemplo: '`@findex @sortas{\} \ @r{(literal \ em @code{@@math})`' ordena a entrada de índice que isso produz sob barra invertida.

Para reduzir a quantidade de chaves de ordenação que você precisa fornecer explicitamente, você pode optar por ignorar determinados caracteres nas entradas de índice para fins de ordenação. Os caracteres que você pode, atualmente, optar por ignorar são '`\`', '`-`', '`<`' e '`@`', que são ignorados fornecendo-se como argumento para o comando `@ set`, respectivamente, `txiindexbackslashignore`, `txiindexhyphenignore`, `txiindexlessthanignore` e `txiindexatsignignore`. Por exemplo, especificar '`@set txiindexbackslashignore`' faz com que a entrada '`\mathopsup`' no índice para este manual seja ordenada como se fosse '`mathopsup`', de forma que apareça entre as outras entradas que começam com ' M'.

> **Cuidado:** Não use dois pontos em uma entrada de índice. No Info, dois pontos separam o nome da entrada de menu do nome do nó, portanto, dois pontos na própria entrada confundem o Info. Veja Seção 4.9.4 [Partes de Menu], Página 37, para mais informações relativas a estrutura de uma entrada de menu.

11.3 Criando Entradas de Índice

As entradas do índice de conceito consistem de texto. A melhor maneira de escrever um índice é a de elaborar entradas que sejam concisas, mas claras. Se você puder fazer isso, o índice geralmente parecerá melhor se as entradas forem escritas exatamente como apareceriam no meio de uma frase, ou seja, colocando em maiúscula somente nomes próprios e siglas que sempre

exigem letras maiúsculas. Essa é a convenção de caso que nós usamos na maioria dos índices dos manuais GNU.

Se você não sabe como tornar uma entrada concisa, mas clara, torne-a mais longa e clara—não concisa e confusa. Se muitas das entradas tiverem várias palavras, o índice poderá parecer melhor se você usar uma convenção diferente: colocar em maiúscula a primeira palavra de cada entrada. Qualquer que seja a convenção de caso que você usar, use-a de consistentemente.

Em qualquer caso, nunca coloque em maiúscula um nome que diferencia maiúsculas de minúsculas, como um nome de função da C ou Lisp ou um comando de shell; isso seria um erro ortográfico. As entradas em índices diferentes do índice de conceito são nomes de símbolos em linguagens de programação ou nomes de programas; esses nomes geralmente diferenciam maiúsculas de minúsculas, de forma que use letras maiúsculas e minúsculas conforme exigido.

É uma boa ideia tornar as entradas do índice exclusivas sempre que possível. Dessa forma, as pessoas que usam a saída gerada impressa ou o preenchimento on-line das entradas do índice não enxergarão listas indiferenciadas. Considere essa uma oportunidade para tornar mais específicas entradas de índice idênticas, de forma que os(as) leitores(as) consigam encontrar mais facilmente o lugar exato que procuram.

Quando você estiver criando entradas de índice, é uma boa prática pensar nas diferentes maneiras pelas quais as pessoas podem procurar algo. Pessoas diferentes *não* pensam nas mesmas palavras quando procuram algo. Um índice útil terá itens indexados sob todas as palavras diferentes que as pessoas possam usar. Por exemplo, um(a) leitor(a) pode pensar que é óbvio que os nomes de duas letras para índices deveriam ser listados sob "Índices, nomes de duas letras, uma vez que "Índices" são o conceito geral. Todavia, outro(a) leitor(a) pode lembrar-se do conceito específico de nomes de duas letras e procurar a entrada listada como "Nomes de duas letras para índices". Um bom índice terá ambas as entradas e ajudará ambos(as) os(as) leitores(as).

Assim como a composição tipográfica, a construção de um índice é uma arte especializada, cujas sutilezas possivelmente não sejam apreciadas até que você precise fazê-la você mesmo(a).

11.4 Imprimindo Índices e Menus

Imprimir um índice significa incluí-lo como parte de um manual ou arquivo do Info. Isso não acontece automaticamente apenas porque você usa `@cindex` ou outros comandos de geração de entrada de índice no arquivo Texinfo; esses apenas faz com que os dados brutos do índice sejam acumulados. Para gerar um índice, você precisa incluir o comando `@printindex` no local no documento onde deseja que o índice apareça. Além disso, como parte do processo de criação de um manual impresso, você precisa executar um programa chamado `texindex` (veja Capítulo 19 [Impresso], Página 150) para ordenar os dados brutos para produzir um arquivo de índice ordenado. O arquivo de índice ordenado é o que realmente é usado para imprimir o índice.

Texinfo oferece seis tipos de índices predefinidos, que são suficientes na maioria dos casos. Veja Capítulo 11 [Índices], Página 89, para informações a respeito disso, bem como para definir teus próprios novos índices, combinar índices e, o mais importante, conselhos sobre escrever as reais entradas do índice. Esta seção foca na impressão de índices, o que é feito com o comando `@printindex`.

`@printindex` recebe um argumento, uma abreviatura de índice de duas letras. Ele lê o correspondente arquivo ordenado de índice (para saída gerada impressa) e o formata apropriadamente em um índice.

O comando `@printindex` não gera um título de capítulo para o índice, pois manuais diferentes tem necessidades diferentes. Consequentemente, você deveria preceder o comando `@printindex` com um adequado comando de seção ou capítulo (geralmente `@appendix` ou `@unnumbered`) para

fornecer o título do capítulo e colocar o índice no sumário. Preceda o título do capítulo com uma linha `@node` como de costume.

Por exemplo:

```
@node Índice Variável
@unnumbered Índice Variável

@printindex vr

@node Índice de Conceito
@unnumbered Índice de Conceito

@printindex cp
```

Se você tiver mais que um índice, nós recomendamos colocar o índice de conceito por último.

- Na saída gerada impressa, `@printindex` produz um índice tradicional de duas colunas, com líderes de pontos entre os termos do índice e os números das páginas.
- Na saída gerada do Info, `@printindex` produz um menu especial contendo o número da linha da entrada, relativo ao início do nó. Leitores(as) do Info podem usar isso para ir para a linha exata de uma entrada, não apenas para o nó que a contém. (Leitores(as) mais antigos(as) do Info irão apenas para o nó). Aqui está um exemplo:

  ```
  * Primeira entrada de índice:   Topo.   (linha  7)
  ```

  O número real de espaços é variável, para justificar à direita o número da linha; foi reduzido aqui para que a linha caiba no manual impresso.
- Na saída gerada de texto simples, `@printindex` produz o mesmo menu, mas os números das linhas são relativos ao início do arquivo, pois isso é mais conveniente para esse formato.
- Na saída gerada de HTML, `@printindex` produz links para as entradas do índice.
- Na saída gerada XML e Docbook, ele simplesmente registra o índice a ser impresso.

11.5 Combinando Índices

Às vezes você irá querer combinar dois índices, como funções e conceitos, talvez porque tenha poucas entradas suficientes que um índice separado pareceria bobo.

Você poderia colocar funções no índice de conceito escrevendo comandos `@cindex` para elas em vez de comandos `@findex` e produzir um manual consistente imprimindo o índice de conceito com o título 'Índice de Função e Conceito' e não imprimir o 'Índice de Função'; mas esse não é um procedimento robusto. Ele funciona somente se o teu documento nunca for incluído como parte de outro documento que seja projetado para ter um índice de função separado; se o teu documento fosse incluído em tal documento, as funções oriundas do teu documento e aquelas oriundas do outro não terminariam juntas. Além disso, para fazer com que os teus nomes de funções apareçam na fonte correta no índice de conceito, você precisaria colocar cada uma delas entre chaves de `@code`.

11.5.1 `@syncodeindex`: Combinando índices usando `@code`

Quando quiser combinar funções e conceitos em um índice, você deveria indexar as funções com `@findex` e indexar os conceitos com `@cindex` e usar o comando `@syncodeindex` para redirecionar as entradas do índice de função para o índice de conceito.

O comando `@syncodeindex` recebe dois argumentos; eles são o nome do índice para redirecionar e o nome do índice para o qual redirecioná-lo. O modelo se parece com isto:

```
@syncodeindex from to
```

Para esse propósito, para os índices são dados nomes de duas letras:

'`cp`' índice de conceito

‘`fn`’ índice de função

‘`vr`’ índice de variável

‘`ky`’ índice de tecla

‘`pg`’ índice de programa

‘`tp`’ índice de tipo de dados

Escreva um comando `@syncodeindex` antes ou logo depois da linha de fim do cabeçalho no início de um arquivo Texinfo. Por exemplo, para mesclar um índice de função com um índice de conceito, escreva o seguinte:

```
@syncodeindex fn cp
```

Isso fará com que todas as entradas designadas para o índice de função sejam mescladas com o índice de conceito.

Para mesclar ambos, um índice de variáveis e um índice de função, em um índice de conceito, escreva o seguinte:

```
@syncodeindex vr cp
@syncodeindex fn cp
```

O comando `@syncodeindex` coloca todas as entradas oriundas do índice ‘from’ (o índice redirecionado) na fonte `@code`, substituindo qualquer fonte padrão que seja usada pelo índice para o qual as entradas são agora direcionadas. Dessa forma, se você direcionar nomes de funções oriundas de um índice de função para um índice de conceito, todos os nomes de funções serão impressos na fonte `@code` como você esperaria.

11.5.2 `@synindex`: Combinando índices

O comando `@synindex` é quase o mesmo que o comando `@syncodeindex`, exceto que não coloca as entradas do índice ‘from’ na fonte `@code`; em vez disso, coloca-os na fonte romana. Assim, você usa `@synindex` ao mesclar um índice de conceito em um índice de função.

Veja Seção 11.4 [Imprimindo Índices e Menus], Página 91, para informações acerca de imprimir um índice no final de um livro ou criar um menu de índice em um arquivo Info.

11.6 Definindo Novos Índices

Além dos índices predefinidos (veja Seção 11.1 [Índices Predefinidos], Página 89), você pode usar os comandos `@defindex` e `@defcodeindex` para definir novos índices. Esses comandos criam novos comandos @ de indexação com os quais você marca entradas de índice. O comando `@defindex` é usado assim:

```
@defindex name
```

Novos nomes de índice geralmente são palavras de duas letras, como ‘`au`’. Por exemplo:

```
@defindex au
```

Isso define um novo índice, chamado o índice ‘`au`’. Ao mesmo tempo, ele cria um novo comando de indexação, `@auindex`, que você pode usar para criar entradas de índice. Use esse novo comando de indexação da mesma forma que usaria um comando de indexação predefinido.

Por exemplo, aqui está um título de seção seguido por uma entrada de índice de conceito e duas entradas de índice ‘`au`’.

```
@section Semântica Cognitiva
@cindex esquemas de imagem cinestésica
@auindex Johnson, Mark
@auindex Lakoff, George
```

(Evidentemente, ‘`au`’ aqui serve como uma abreviação para “autor”.)

Texinfo constrói o novo comando de indexação concatenando o nome do índice com '`index`'; portanto, definir um índice '`xy`' leva à criação automática de um comando `@xyindex`.

Use o comando `@printindex` para imprimir o índice, como você faz com índices predefinidos. Por exemplo:

```
@node Índice de Autor
@unnumbered Índice de Autor

@printindex au
```

O `@defcodeindex` é como o comando `@defindex`, exceto que, na saída impressa, ele imprime entradas em uma fonte `@code` por padrão, em vez de uma fonte romana.

Você deveria definir novos índices antes da linha de fim de cabeçalho de um arquivo Texinfo e (é claro) antes de quaisquer comandos `@synindex` ou `@syncodeindex` (veja Seção 3.2 [Cabeçalho do Arquivo do Texinfo], Página 15).

Conforme mencionado anteriormente (veja Seção 11.1 [Índices Predefinidos], Página 89), nós recomendamos ter um índice unitário no documento final sempre que possível, independentemente de quantos índices de fonte você usar, pois assim leitores(as) terão somente um lugar para procurar.

Ao criar um índice, TEX cria um arquivo cuja extensão é o nome do índice (veja [Nomes dos arquivos de índice], Página 152). Portanto, você deveria evitar usar nomes de índices que colidam com extensões usadas para outros propósitos, tais como '`.aux`' ou '`.xml`'. `makeinfo` já informa um erro se um novo índice conflitar com um nome de extensão bem conhecido.

12 Inserções Especiais

Texinfo fornece vários comandos para inserir caracteres que tenham significado especial no Texinfo, tais como colchetes, e para outros elementos gráficos que não correspondem a caracteres simples que você pode digitar.

Esses são:

- Os caracteres especiais do Texinfo: '`@ {} , \ #`'.
- Espaço em branco dentro e ao redor de uma frase.
- Acentos.
- Pontos e balas.
- O logotipo TEX e o símbolo de direitos autorais.
- Os símbolos de moeda euro e libras.
- O símbolo de graus.
- O sinal de menos.
- Expressões matemáticas.
- Glifos para exemplos de programação: avaliação, macros, erros, etc.
- Notas de rodapé.

12.1 Caracteres Especiais: Inserindo @ {} , \

'`@`' e chaves são os caracteres especiais básicos no Texinfo. Para inserir esses caracteres de forma que apareçam no texto, você precisa colocar um '`@`' na frente desses caracteres para evitar que o Texinfo os interprete mal. Comandos alfabéticos também são fornecidos.

Os demais caracteres (vírgula, barra invertida, cerquilha) são especiais somente em contextos restritos, conforme explicado nas respectivas seções.

12.1.1 Inserindo '@' com `@@` e `@atchar{}`

`@@` produz um caractere '`@`' na saída gerada. Não coloque colchetes depois de um comando `@@`.

`@atchar{}` também produz um caractere '`@`' na saída gerada. Ele precisa de colchetes seguintes, como de costume para comandos alfabéticos. Em Condicionais inline (veja Seção 16.4 [Condicionais Inline], Página 131), pode ser necessário evitar usar o caractere literal '`@`' no fonte (e possivelmente seja mais claro em outros contextos).

12.1.2 Inserindo '{ '}' com `@{ @}` e `@l rbracechar{}`

`@{` produz um '`{`' na saída gerada e `@}` produz um '`}`'. Não coloque colchetes depois de um comando `@{` ou `@}`.

`@lbracechar{}` e `@rbracechar{}` também produzem caracteres '`{`' e '`}`' unitários na saída gerada. Eles precisam de colchetes seguintes, como de costume para comandos alfabéticos. Em Condicionais inline (veja Seção 16.4 [Condicionais Inline], Página 131), pode ser necessário evitar usar caracteres de colchetes literais no fonte (e possivelmente seja mais claro em outros contextos).

12.1.3 Inserindo ',' com `@comma{}`

Normalmente, uma vírgula ',' é um caractere normal que pode ser simplesmente digitado em tua entrada onde você precisar dela.

No entanto, Texinfo usa a vírgula como um caractere especial somente em um contexto: para separar argumentos para aqueles comandos do Texinfo, como `@acronym` (veja Seção 7.1.14 [`@acronym`], Página 62) e `@xref` (veja Capítulo 6 [Referências Cruzadas], Página 45), bem como

macros definidas por usuário(a) (veja Seção 17.1 [Definindo Macros], Página 137), que levam mais que um argumento. Como um caractere de vírgula confundiria a análise do Texinfo para esses comandos, você precisa usar o comando '`@comma{}`' se quiser passar uma vírgula real. Aqui estão alguns exemplos:

```
@acronym{ABC, Uma @comma{} bizarra}
@xref{Vírgula,, O símbolo @comma{}}
@mymac{Um argument@comma{} contendo uma vírgula}
```

Embora '`@comma{}`' possa ser usada em praticamente qualquer lugar, não existe necessidade dela em nenhum lugar, exceto nesse caso incomum.

(Incidentalmente, o nome '`@comma`' carece do sufixo '`char`' usado nos comandos complementares dele somente por razões históricas. Não pareceu importante o suficiente para definir um sinônimo).

12.1.4 Inserindo '\' com `@backslashchar{}`

Normalmente, uma barra invertida '\' é um caractere normal no Texinfo que pode ser simplesmente digitado em tua entrada onde você precisar. O resultado é o de compor a barra invertida oriunda da fonte da máquina de escrever.

Porém, Texinfo usa a barra invertida como um caractere especial em um contexto restrito: para delimitar argumentos formais nos corpos de macros definidas por usuário(a) (veja Seção 17.1 [Definindo Macros], Página 137).

Devido aos caprichos da análise de argumentos de macro, é mais confiável passar um comando alfabético que produza uma barra invertida em vez de usar uma `\` literal. Daí `@backslashchar{}`. Aqui está um exemplo de chamada de macro:

```
@mymac{Um argument@backslashchar{} com uma barra invertida}
```

Documentos do Texinfo também podem usar `\` como um caractere de comando dentro de `@math` (veja Seção 12.7 [Inserindo Fórmulas Matemáticas], Página 102). Nesse caso, `@\` ou `\backslash` produz uma barra invertida "matematicista" (proveniente da fonte do símbolo matemático), enquanto `@backslashchar{}` produz uma barra invertida de máquina de escrever como sempre.

Embora '`@backslashchar{}`' possa ser usado em praticamente qualquer lugar, não existe necessidade dele, exceto nesses casos incomuns.

12.1.5 Inserindo '#' com `@hashchar{}`

Normalmente, uma cerquilha '#' é um caractere normal no Texinfo que pode ser simplesmente digitado em tua entrada onde você precisar. O resultado é o de compor o caractere cerquilha oriundo da fonte atual.

Esse caractere tem muitos outros nomes, variando de acordo com a localidade, como "sinal numérico", "libra" e "octothorp". Às vezes também é chamado de "sustenido" ou "sinal sustenido", uma vez que se assemelha vagamente ao símbolo musical com esse nome. Em situações onde Texinfo é usado, "cerquilha" é o mais comum em nossa experiência.

No entanto, Texinfo usa o caractere cerquilha como um caractere especial em um contexto restrito: para introduzir a chamada diretiva `#line` e variantes (veja Seção 17.6 [Processadores Externos de Macro], Página 143).

Portanto, para a finalidade de compor um caractere cerquilha real em tal local (por exemplo, em um programa que precisa de documentação acerca de `#line`), é necessário usar `@hashchar{}` ou alguma outra construção. Aqui está um exemplo:

```
@hashchar{} 10 "exemplo.c"
```

Embora '`@hashchar{}`' possa ser usado em praticamente qualquer lugar, não existe necessidade dele em nenhum lugar, exceto nesse caso incomum.

12.2 Inserindo Caracteres de Citação

Conforme explicado na seção anterior acerca de convenções gerais de entrada do Texinfo (veja Seção 2.1 [Convenções], Página 9), arquivos fonte do Texinfo usam o caractere ASCII ‘ (decimal 96) para produzir uma aspa esquerda (‘), e o ASCII ’ (decimal 39) para produzir uma aspa direita (’). Duplicar esses caracteres de entrada (‘‘ e ’’) produz aspas duplas (“ e ”). Essas são as convenções usadas pelo TEX.

Isso funciona bem para texto. No entanto, em exemplos de código de computador, leitores(as) são especialmente propensos(as) a recortar e colar o texto literalmente—e, infelizmente, alguns visualizadores de documentos estragarão esses caracteres. (O leitor livre de PDF `xpdf` funciona bem, mas outros leitores de PDF, tanto livres quanto não livres, tem problemas).

Se isso for uma preocupação para você, Texinfo fornece estes dois comandos:

`@codequoteundirected` *on-off*
: faz com que a saída gerada para o caractere ’ em ambientes de código seja a aspa simples indirecionada, como esta:
'.

`@codequotebacktick` *on-off*
: faz com que a saída gerada para o caractere ‘ em ambientes de código seja o caractere crase (acento grave autônomo), assim:
`.

Se você quiser essas configurações somente para parte do documento, `@codequote... off` restaurará o comportamento normal, como em `@codequoteundirected off`.

Essas configurações afetam `@code`, `@example`, `@kbd`, `@samp`, `@verb` e `@verbatim`. Veja Seção 7.1.1 [Realçamento Útil], Página 56.

Esse recurso costumava ser controlado usando-se `@set` para mudar os valores das variáveis correspondentes `txicodequoteundirected` e `txicodequotebacktick`; elas ainda são suportadas, mas a interface de comando é preferida.

12.3 Inserindo Espaço

As seções a seguir descrevem comandos que controlam espaçamentos de vários tipos dentro e depois de frases.

12.3.1 Espaços Múltiplos

Normalmente, vários caracteres de espaço em branco (espaço, tabulação e nova linha) são recolhidos em um espaço.

Ocasionalmente, você pode querer produzir vários espaços consecutivos, seja para fins de exemplo (por exemplo, o que teu programa faz com vários espaços como entrada), ou meramente para fins de aparência em títulos ou listas. Texinfo suporta três comandos: `@`*`SPACE`*, `@`*`TAB`* e `@`*`NL`*, todos os quais inserem um espaço na saída gerada. (Aqui, `@`*`SPACE`* representa um caractere ‘`@`’ seguido por um espaço, ou seja, ‘`@ `’, *`TAB`* representa um caractere de tabulação, e `@`*`NL`* representa um caractere ‘`@`’ e fim de linha, ou seja, quando ‘`@`’ é o último caractere em uma linha).

Por exemplo,

```
Exemplo@ @ @ @
espaçoso.
```

produz

```
Exemplo    espaçoso.
```

Outros usos possíveis de `@`*`SPACE`* foram subsumidos por `@multitable` (veja Seção 9.5 [Tabelas Multi Colunas], Página 80).

Não coloque chaves depois de nenhum desses comandos.

Para produzir um espaço inquebrável, veja-se Seção 13.6 [@tie], Página 112.

12.3.2 Não Finalizando Uma Frase

Quando um ponto, ponto de exclamação ou ponto de interrogação estão no final de uma frase, um pouco mais de espaço é inserido depois deles em um manual tipográfico.

Geralmente, Texinfo consegue determinar automaticamente quando um ponto termina uma frase. No entanto, comandos especiais são necessários em algumas circunstâncias. Use o comando `@:` depois de um ponto, ponto de interrogação, ponto de exclamação ou dois pontos que não deveriam ser seguidos por espaço extra. Isso é necessário nas seguintes situações:

1. Depois de um ponto que finalize uma abreviação em minúscula que não esteja no final de uma frase.
2. Quando uma observação entre parênteses no meio de uma frase (como esta!) terminar com um ponto, ponto de exclamação ou ponto de interrogação, `@:` deveria ser usado depois do parêntese direito. Da mesma forma para colchetes direitos e aspas direitas (ambas, simples e duplas).

Por exemplo:

```
‘foo vs.@: bar (ou?)@: baz’,
```

A primeira linha abaixo mostra a saída gerada e, para comparação, a segunda linha mostra o espaçamento quando os comandos ‘`@:`’ não foram usados.

foo vs. bar (ou?) baz
foo vs. bar (ou?) baz

Se você olhar atentamente, verá um pouco de espaço não essencial depois de ‘`vs.`’ e ‘`(ou?)`’.

Possivelmente te ajude lembrar o que `@:` faz imaginando que ele representa um caractere minúsculo invisível que impede que uma palavra termine com um ponto final.

Alguns comandos do Texinfo forçam espaçamento normal entre palavras, de forma que você não tenha que inserir `@:` onde normalmente faria. Esses são os comandos de realçamento semelhantes a código, `@var`, `@abbr` e `@acronym` (veja Seção 7.1.1 [Realçamento Útil], Página 56). Por exemplo, em ‘`@code{foo. bar}`’ o ponto não é considerado ser o fim de uma frase, e nenhum espaço extra é inserido.

`@:` não tem efeito na saída HTML ou Docbook.

12.3.3 Finalizando Uma Frase

Como mencionado acima, Texinfo normalmente insere espaço adicional depois do fim de uma frase. Ele usa a mesma heurística para isso que o TEX: uma frase termina com um ponto final, ponto de exclamação ou ponto de interrogação, precedido ou seguido por pontuação opcional de fechamento, e, então, espaço em branco, e *não* precedido por uma letra maiúscula.

Use `@.` em vez de um ponto, `@!` em vez de um ponto de exclamação e `@?` em vez de um ponto de interrogação ao final de uma frase que termine com uma letra maiúscula. Não coloque chaves depois de quaisquer desses comandos. Por exemplo:

```
Dê para M.I.B. e para M.E.W@. Também, dê para R.J.C@.
Dê para M.I.B. e para M.E.W. Também, dê para R.J.C.
```

A saída gerada segue. Na saída gerada impressa e no Info, você pode ver o espaço em branco extra desejado depois do ‘`W`’ na primeira linha.

Dê para M.I.B. e para M.E.W. Também, dê para R.J.C.
Dê para M.I.B. e para M.E.W. Também, dê para R.J.C.

Na saída gerada de HTML, `@.` é equivalente a um simples ‘`.`’; o mesmo vale para `@!` e `@?`.

A "pontuação de fechamento" mencionada acima é definida como um parêntese direito (')'), colchete direito (']', ou aspa direita, simples ou dupla ('' e ''''; as muitas aspas direitas Unicode adicionais possíveis não estão incluídas). Esses caracteres podem ser considerados como invisíveis com respeito a se um determinado ponto finaliza uma frase. (Essa é a mesma regra que TEX). Por exemplo, os pontos em 'foo.) Bar' e 'foo.'' Bar' finalizam frases.

Os significados de @: e @., etc. no Texinfo são projetados para funcionar bem com os comandos de movimento de frase do Emacs (veja Seção "Sentences" em *O Manual do GNU Emacs*). Possivelmente ajude imaginar que o '@' em '@.', etc., é uma letra minúscula invisível 'a' que torna uma letra maiúscula antes dela irrelevante para os propósitos de decidir se o ponto final termina a frase.

Alguns comandos do Texinfo não são considerados como sendo uma abreviação, mesmo que possam terminar com uma letra maiúscula quando expandidos, de forma que você não tenha que inserir @. e acompanhantes. Notavelmente, esse é o caso para comandos de realçamento semelhantes a código, argumentos @var terminando com uma letra maiúscula, @LaTeX e @TeX. Por exemplo, essa frase terminou com '... @code{@@TeX}.'; @. não foi necessário. Da mesma forma, em ... @var{VARNAME}. Texto o ponto depois de *VARNAME* termina a frase; não existe necessidade de usar @..

12.3.4 @frenchspacing *val*: Controle de Espaçamento de Frase

Na tipografia americana, é tradicional e correto colocar espaço extra ao final de uma frase. Esse é o padrão no Texinfo (implementado no Info e na saída gerada impressa; para HTML, nós não tentamos substituir o navegador). Na tipografia francesa (e outras), esse espaço extra é errado; todos os espaços são uniformes.

Portanto, Texinfo fornece o comando @frenchspacing para controlar o espaçamento depois da pontuação. Ele lê o resto da linha como argumento, que precisa ser a única palavra 'on' ou 'off' (sempre essas palavras, independentemente do idioma do documento). Aqui está um exemplo:

```
@frenchspacing on
Isso é texto. Duas frases. Três frases. Espaçamento francês.

@frenchspacing off
Isso é texto. Duas frases. Três frases. Espaçamento não francês.
```

produz:

Isso é texto. Duas frases. Três frases. Espaçamento francês.

Isso é texto. Duas frases. Três frases. Espaçamento não francês.

@frenchspacing também afeta a saída gerada depois de @., @! e @? (veja Seção 12.3.3 [Finalizando Uma Frase], Página 98).

@frenchspacing não tem efeito na saída gerada de HTML ou Docbook; para XML, ele gera uma transliteração dele mesmo (veja Seção 1.2 [Formatos de Saída], Página 4).

12.3.5 @dmn{*dimension*}: Formatar uma dimensão

Você pode usar o comando @dmn para formatar uma dimensão com um pouco de espaço extra na saída gerada impressa. Ou seja, ao ver @dmn, TEX insere apenas espaço suficiente para a composição adequada; em outros formatos da saída gerada, os comandos de formatação não inserem espaço algum.

Para usar o comando @dmn, escreva o número e depois siga-o imediatamente, sem espaço intermediário, por @dmn, e depois pela dimensão entre chaves. Por exemplo,

```
Papel A4 tem 8,27@dmn{in} de largura.
```

produz

Papel A4 tem 8,27 in de largura.

Nem todo mundo usa esse estilo. Algumas pessoas preferem '8.27 in.' ou '8.27 inches'. Nesses casos, no entanto, você precisa usar `@tie` (veja Seção 13.6 [`@tie`], Página 112) ou `@w` (veja Seção 13.5 [`@w`], Página 111) de forma que nenhuma quebra de linha possa ocorrer entre o número e a dimensão. Além disso, se você escrever um ponto depois de uma abreviação dentro de uma frase (como com 'in.' acima), você deveria escrever '`@:`' depois do ponto para evitar que TEX insira espaços em branco extras, como mostrado aqui. Veja Seção 12.3.2 [Não Finalizando Uma Frase], Página 98.

12.4 Inserindo Acentos

Aqui está uma tabela com os comandos que Texinfo fornece para inserir acentos flutuantes. Eles todos precisam de um argumento, o caractere a ser acentuado, que pode ou ser fornecido entre chaves como de costume (`@'{e}`), ou, como um caso especial, as chaves podem ser omitidas, em cujo caso o argumento é o próximo caractere (`@'e`). Isso é para tornar o fonte o mais conveniente possível para digitar-se e ler-se, já que caracteres acentuados são muito comuns em alguns idiomas.

Se o comando for alfabético, como `@dotaccent`, então precisa existir um espaço entre o nome do comando e o argumento se chaves não forem usadas. Se o comando for não alfabético, como `@'`, então precisa *não* existir um espaço; o argumento é o exato próximo caractere.

Exceção: o argumento para `@tieaccent` precisa ser colocado entre chaves (já que são dois caracteres em vez de um).

Para obter a saída gerada dos verdadeiros caracteres acentuados no Info, não apenas as transliterações ASCII, é necessário especificar-se `@documentencoding` com uma codificação que suporte os caracteres exigidos (veja Seção 15.2 [`@documentencoding`], Página 126). Nesse caso, você também pode usar caracteres não ASCII (por exemplo, pré-acentuados) no arquivo fonte.

Comando	**Saída gerada**	**O quê**
`@"o`	ö	acento trema
`@'o`	ó	acento agudo
`@,{c}`	ç	acento cedilha
`@=o`	ō	acento macron/overbar
`@^o`	ô	acento circunflexo
`@`o`	ò	acento grave
`@~o`	õ	acento til
`@dotaccent{o}`	ȯ	acento overdot
`@H{o}`	ő	longo trema húngaro
`@ogonek{a}`	ą	ogonek
`@ringaccent{o}`	o̊	acento ring
`@tieaccent{oo}`	o͡o	acento de ligação posterior
`@u{o}`	ŏ	acento breve
`@ubaraccent{o}`	o̲	acento underbar
`@udotaccent{o}`	ọ	acento underdot
`@v{o}`	ǒ	acento caron/hacek/check

Esta tabela lista os comandos do Texinfo para inserir outros caracteres comumente usados em idiomas diferentes do inglês.

`@exclamdown{}`	¡	! de cabeça para baixo
`@questiondown{}`	¿	? de cabeça para baixo

`@aa{} @AA{}`	å Å	a,A com círculo
`@ae{} @AE{}`	æ Æ	ligaduras ae,AE
`@dh{} @DH{}`	ð Ð	eth islandês
`@dotless{i}`	ı	i sem ponto
`@dotless{j}`	ȷ	j sem ponto
`@l{} @L{}`	ł Ł	L,l suprimido
`@o{} @O{}`	ø Ø	O,o com barra
`@oe{} @OE{}`	œ Œ	ligaduras de oe,OE
`@ordf{} @ordm{}`	a º	Ordinais espanhóis
`@ss{}`	ß	es-zet ou sharp S
`@th{} @TH{}`	þ Þ	thorn islandês

12.5 Inserindo Aspas

Use caracteres de aspas simples duplicadas para iniciar e terminar citações: ``...''. TEX converte duas aspas simples para marcas duplicadas de citação à esquerda e à direita, “assim”, e Info converte caracteres duplicados de aspas simples para aspas duplas ASCII: ``...'' se torna "...".

Ocasionalmente, você pode precisar produzir duas aspas simples consecutivas; por exemplo, ao documentar uma linguagem de computador como Maxima, onde '' é um comando válido. Você pode fazer isso com a entrada `'@w{}'`; o comando vazio `@w` interrompe a combinação nos caracteres de aspas duplas.

O caractere de aspa esquerda (`, código ASCII 96) usado no Texinfo é um acento grave nos padrões de conjunto de caracteres ANSI e ISO. Nós o usamos como um caractere de aspas porque assim é como TEX está configurado, por padrão.

Texinfo suporta várias outras aspas usadas em idiomas diferentes do inglês. Abaixo está uma tabela com os comandos que Texinfo fornece para inserir aspas.

Para a finalidade de se obter os símbolos para as aspas na saída gerada codificada do Info, é necessário especificar-se `@documentencoding UTF-8`. (Veja Seção 15.2 [`@documentencoding`], Página 126). Guillemets duplos também estão presentes na ISO 8859-1 (também conhecida como Latin 1) e na ISO 8859-15 (também conhecida como Latin 9).

As fontes padrão do TEX suportam as aspas usuais usadas em inglês (aquelas produzidas com aspas simples de ASCII unitárias e duplicadas). Para as outras aspas, TEX usa fontes European Computer Modern (EC) (`ecrm1000` e outras variantes). Essas fontes estão disponíveis gratuitamente, é claro; você pode baixá-las a partir de `http://ctan.org/pkg/ec`, entre outros lugares.

As fontes livres EC são fontes bitmap criadas com Metafont. Especialmente para visualização on-line, as versões Type 1 (vetor) das fontes são preferíveis; elas estão disponíveis no pacote de fontes CM-Super (`http://ctan.org/pkg/cm-super`).

Ambas as distribuições incluem instruções de instalação.

Comando	**Glifo**	**Nome Unicode (ponto)**
`@quotedblleft{} ```	“	Aspas duplas esquerdas (U+201C)
`@quotedblright{} ''`	”	Aspas duplas direitas (U+201D)
`@quoteleft{} ` `	‘	Aspas simples esquerdas (U+2018)
`@quoteright{} '`	’	Aspas simples direitas (U+2019)
`@quotedblbase{}`	„	Aspas duplas baixo-9 (U+201E)
`@quotesinglbase{}`	‚	Aspas simples baixo-9 (U+201A)

`@guillemetleft{}`	«	Aspas duplas apontando para a esquerda (U+00AB)
`@guillemetright{}`	»	Aspas duplas apontando para a direita (U+00BB)
`@guilsinglleft{}`	‹	Aspas simples apontando para a esquerda (U+2039)
`@guilsinglright{}`	›	Aspas simples apontando para a direita (U+203A)

Para as aspas de ângulo duplo, nomes de glifos da Adobe e LaTeX também são suportados: `@guillemotleft` e `@guillemotright`. Esses nomes estão incorretos; um "guillemot" é uma espécie de pássaro (um tipo de auk).

As tradições para o uso de aspas variam muito entre os idiomas (`http://en.wikipedia.org/wiki/Quotation_mark`). Texinfo não fornece comandos ou configurações para composição de aspas de acordo com as numerosas tradições. Portanto, você tem que escolher os comandos apropriados para o idioma do teu manual. Às vezes, apelidos (veja Seção 17.4 [`@alias`], Página 142) podem simplificar o uso e tornar o código-fonte mais legível. Por exemplo, em alemão, `@quotedblbase` é usado para as aspas duplas à esquerda, e as aspas duplas à direita são o glifo produzido por `@quotedblleft`, o que é contraintuitivo. Portanto, nesse caso, os seguintes apelidos seriam convenientes:

```
@alias lgqq = quotedblbase
@alias rgqq = quotedblleft
```

12.6 `@sub` e `@sup`: Inserindo Subscritos e Sobrescritos

Você consegue inserir subscritos e sobrescritos, em texto ou matemática, com os comandos `@sub` e `@sup`. (Para outras expressões matemáticas, veja-se a próxima seção). Por exemplo, aqui está um subscrito e sobrescrito puramente textual:

```
aqui@sub{abaixo}@sup{acima}
```

produz:

aqui$_{\text{abaixo}}{}^{\text{acima}}$

Dentro de `@math`, `@sub` e `@sup` produzem subscritos e sobrescritos matemáticos. Isso usa uma fonte diferente na saída gerada do TeX (itálico de matemática, em vez de itálico de texto); não faz diferença nos outros formatos de saída. Aqui está um exemplo:

```
@math{e@sup{x}}
```

produz:

e^x

No Info e texto simples, independentemente de ser usado dentro de `@math`, `@sub{`*texto*`}` é gerado como '`_{`*texto*`}`' e `@sup{`*texto*`}` como '`^{`*texto*`}`', incluindo as chaves literais (para marcar o início e o fim do texto de "script" para o(a) leitor(a)).

Quando o formato da saída gerada (e o programa de exibição) permite (matemática do TeX, HTML), o sobrescrito é configurado acima do subscrito quando ambos os comandos forem fornecidos consecutivamente.

12.7 `@math`: Inserindo Expressões Matemáticas

Você consegue escrever uma expressão matemática curta com o comando `@math`. Escreva a expressão matemática entre chaves, assim:

```
@math{(a + b) = (b + a)}
```

Isso produz o seguinte no TeX:

$(a + b) = (b + a)$

e o seguinte em outros formatos:

```
(a + b) = (b + a)
```

O comando `@math` não tem efeito especial na saída gerada de Info e de HTML. `makeinfo` expande quaisquer comandos @ como de costume, mas não tenta usar (ou produzir) boa formatação matemática de forma alguma (sem uso de MathML, etc.). A saída gerada de HTML é delimitada por `<em>...</em>`, mas nada mais.

No entanto, no que diz respeito à saída gerada do TEX, comandos matemáticos simples do TEX são permitidos no `@math`, começando com '\'. Em essência, o `@math` comuta para o modo matemático simples do TEX. (Exceção: o comando simples do TEX `\sup`, que tipografa o nome do operador matemático 'sup', precisa ser acessado como `\mathopsup`, devido ao conflito com o comando `@sup` do Texinfo).

Isso permite que você use todas as sequências de controle matemático simples do TEX para símbolos, funções e assim por diante, e assim obtenha a formatação adequada na saída gerada do TEX, pelo menos.

Os comandos `@sub` e `@sup` descritos na seção anterior produzem subscritos e sobrescritos na saída gerada de HTML, bem como no TEX; os caracteres simples do TEX `_` e `^` para subscritos e sobrescritos são reconhecidos pelo TEX dentro do `@math`, mas não fazem nada de especial em HTML ou outros formatos de saída.

É melhor usar '\' em vez de '`@`' para quaisquer comandos matemáticos desse tipo; caso contrário, `makeinfo` reclamará. Por outro lado, `makeinfo` permite entrada com chaves correspondentes (inescapadas), como '`k_{75}`'; ele reclama acerca dessas chaves vazias na entrada regular.

Aqui está um exemplo:

```
@math{\sin 2\pi \equiv \cos 3\pi}
```

que se parece com isto no TEX:

$$\sin 2\pi \equiv \cos 3\pi$$

mas que se parece com a entrada no Info e em HTML:

```
\sin 2\pi \equiv \cos 3\pi
```

Como '\' é um caractere de escape dentro de `@math`, você pode usar `@\` para obter uma barra invertida literal (`\\` funcionará no TEX, mas você obteria os dois caracteres literais '\\' no Info). `@\` não está definido fora de `@math`, pois uma '\' normalmente produz uma '\' literal (máquina de escrever). Você também pode usar `@backslashchar{}` em qualquer modo para obter uma barra invertida de máquina de escrever. Veja Seção 12.1.4 [Inserindo uma Barra Invertida], Página 96.

Para equações exibidas, você precisa, no presente, usar TEX diretamente (veja Seção 16.3 [Comandos do Formatador Bruto], Página 130).

12.8 Glifos para Texto

Texinfo tem suporte para alguns glifos adicionais que são comumente usados em texto impresso, mas não disponíveis em ASCII. Claro, existem muitos milhares mais. É possível usar caracteres Unicode como estão no que diz respeito ao `makeinfo`, mas o TEX não é tão sortudo.

12.8.1 `@TeX{}` (TEX) e `@LaTeX{}` (LATEX)

Use o comando `@TeX{}` para gerar 'TEX'. Em um manual impresso, esse é um logotipo especial que é diferente de três letras comuns. No Info, ele se parece apenas com '`TeX`'.

Da mesma forma, use o comando `@LaTeX{}` para gerar 'LATEX', que é ainda mais especial em manuais impressos (e diferente do incorreto `La@TeX{}`. No Info, o resultado é apenas '`LaTeX`'.

(LaTeX é outro pacote de macro criado com fundamento no TeX, muito vagamente análogo ao Texinfo, pois enfatiza a estrutura lógica, mas muito (muito) maior).

A grafia desses comandos é incomum no Texinfo, pois eles usam ambas, letras maiúsculas e minúsculas.

12.8.2 `@copyright{}` (©)

Use o comando `@copyright{}` para gerar o símbolo de copyright, '©'. Onde possível, esse é um '`c`' dentro de um círculo; no Info, isso é '`(C)`'.

Juridicamente, não é necessário usar o símbolo de direitos autorais; a palavra em inglês 'Copyright' é suficiente, de acordo com tratado internacional.

12.8.3 `@registeredsymbol{}` (®)

Use o comando `@registeredsymbol{}` para gerar o símbolo de Registrado, '®'. Onde possível, esse é um '`R`' dentro de um círculo; no Info, isso é '`(R)`'.

12.8.4 `@dots{}` (...) e `@enddots{}` (....)

Uma *reticência* (uma sequência de pontos) seria espaçada incorretamente quando tipografada como uma sequência de pontos, de forma que um comando especial é usado no Texinfo: use o comando `@dots{}` para gerar uma reticência normal, que são três pontos em uma linha, apropriadamente espaçados ... assim. Para enfatizar: não escreva simplesmente três pontos no arquivo de entrada; isso funcionaria para a saída gerada do arquivo do Info, mas produziria a quantidade errada de espaço entre os pontos no manual impresso. O comando `@enddots{}` gera reticências no final da frase, que também tem três pontos, mas com espaçamento diferente depois, ... Observe atentamente para ver a diferença.

Aqui está uma reticência: .... Aqui estão três pontos em uma linha: ... .

Na saída gerada impressa (e geralmente em HTML), os três pontos em uma linha estão muito mais próximos um ao outro que os pontos nas reticências.

12.8.5 `@bullet` (•)

Use o comando `@bullet{}` para gerar um ponto redondo grande, ou o mais próximo possível de um. No Info, um asterisco é usado. Aqui está um marcador: •

Ao usar `@bullet` em `@itemize`, você não precisa digitar as chaves, porque `@itemize` as fornece. (veja Seção 9.2 [`@itemize`], Página 75).

12.8.6 `@euro` (): Símbolo da Moeda Euro

Use o comando `@euro{}` para gerar ''. Onde possível, esse é o símbolo da moeda Euro. Caso contrário, a palavra '`Euro`' é usada.

Texinfo não pode magicamente sintetizar suporte para o símbolo do Euro onde o sistema subjacente (fontes, software, o que for) não o suporta. Portanto, você pode achar preferível usar a palavra "Euro". (Em contextos bancários, a abreviação para o Euro é EUR).

Para a finalidade de obter o símbolo do Euro na saída gerada codificada do Info, por exemplo, é necessário especificar `@documentencoding ISO-8859-15` ou `@documentencoding UTF-8` (Veja Seção 15.2 [`@documentencoding`], Página 126). O símbolo do Euro está na ISO 8859-15 (também conhecida como Latin 9) e *não* está na mais amplamente usada ISO 8859-1 (Latin 1).

O símbolo do Euro não existe nas fontes padrão do TeX (que foram projetadas antes da existência jurídica do Euro). Portanto, o TeX usa uma fonte adicional, chamada `feymr10` (junto com outras variáveis). Ela está disponível gratuitamente, é claro; você consegue baixá-la a partir de `http://ctan.org/pkg/eurosym`, entre outros lugares. A distribuição inclui instruções de instalação.

12.8.7 `@pounds` (£): Libras esterlinas

Use o comando `@pounds{}` para gerar '£'. Onde possível, esse é o símbolo para a Libra esterlina, moeda britânica. Caso contrário, é '`#`'.

12.8.8 `@textdegree` (°): símbolo de Graus

Use o comando `@textdegree{}` para gerar '°'. Onde possível, esse é o símbolo normal para Graus. Caso contrário, é um '`o`'.

12.8.9 `@minus` (−): Inserindo um Sinal de Menos

Use o comando `@minus{}` para gerar um sinal de menos. Em uma fonte de largura fixa, isso é um hífen, mas em uma fonte proporcional, o símbolo tem o comprimento habitual para um sinal de menos—um pouco mais longo que um hífen, mais curto que um travessão:

'−' é um sinal de menos gerado com '`@minus{}`',

'-' é um hífen gerado com o caractere '`-`',

'—' é um travessão para texto.

Na fonte de largura fixa usada pelo Info, `@minus{}` é o mesmo que um hífen.

Você deveria não usar `@minus{}` dentro de `@code` ou `@example` porque a distinção de largura não é feita na fonte de largura fixa que eles usam.

Ao usar `@minus` para especificar a marca que inicia cada entrada em uma lista de itens, você não precisa digitar as chaves (veja Seção 9.2 [`@itemize`], Página 75).

Se você realmente quer tipografar alguma matemática que faça uma subtração, é melhor usar `@math`. Então o caractere regular '`-`' produz um sinal de menos, como em `@math{a-b}` (veja Seção 12.7 [Inserindo Fórmulas Matemáticas], Página 102).

12.8.10 `@geq` (≥) e `@leq` (≤): Inserindo Relações

Use os comandos `@geq{}` e `@leq{}` para gerar sinais de maior/menor que ou igual a, '≥' e '≤'. Quando esses símbolos não estão disponíveis, as sequências ASCII '`>=`' e '`<=`' são geradas.

12.9 Glifos para Programação

No Texinfo, o código frequentemente é ilustrado em exemplos que são delimitados por `@example` e `@end example`, ou por `@lisp` e `@end lisp`. Em tais exemplos, você pode indicar os resultados da avaliação ou uma expansão usando '⇒' ou '↦'. Da mesma forma, existem comandos para inserir glifos para indicar saída gerada impressa, mensagens de erro, equivalência de expressões, o local do ponto em um editor e sequências de operação da GUI.

Os comandos de inserção de glifos não precisam ser usados dentro de um exemplo, mas na maioria das vezes são. Todos os comandos de inserção de glifos são seguidos por chaves vazias.

12.9.1 Sumário de Glifos

Aqui está um resumo dos comandos de glifo:

⇒	`@result{}` indica o resultado de uma expressão.
↦	`@expansion{}` indica os resultados de uma expansão de macro.
⊣	`@print{}` indica saída impressa.
error	`@error{}` indica que o texto seguinte é uma mensagem de erro.
≡	`@equiv{}` indica a equivalência exata de duas formas.

⋆ `@point{}` mostra o local do ponto.

A → B `@clicksequence{A @click{} B` indica uma sequência de operação da GUI: primeiro A, depois clicar em B, ou escolher B em um menu, ou então selecioná-lo.

12.9.2 `@result{}` (⇒): Resultado de uma Expressão

Use o comando `@result{}` para indicar o resultado da avaliação de uma expressão.

O comando `@result{}` é exibido como '⇒', uma seta com haste dupla ou (quando essa não estiver disponível) como a sequência ASCII '`=>`'.

Assim, o seguinte,

```
(cdr '(1 2 3))
    ⇒ (2 3)
```

pode ser lido como "`(cdr '(1 2 3))` avalia para `(2 3)`".

12.9.3 `@expansion{}` (↦): Indicando uma Expansão

Quando uma expressão é uma chamada de macro, ela se expande para uma nova expressão. Você pode indicar o resultado da expansão com o comando `@expansion{}`.

O comando `@expansion{}` é exibido como '↦', uma seta longa com uma base plana ou (quando essa não estiver disponível) como a sequência ASCII '`==>`'.

Por exemplo, o seguinte

```
@lisp
(third '(a b c))
    @expansion{} (car (cdr (cdr '(a b c))))
    @result{} c
@end lisp
```

produz

```
(third '(a b c))
    ↦ (car (cdr (cdr '(a b c))))
    ⇒ c
```

que pode ser lido como:

> `(third '(a b c))` expande para `(car (cdr (cdr '(a b c))))`; o resultado da avaliação da expressão é `c`.

Frequentemente, como nesse caso, um exemplo aparenta melhor se os comandos `@expansion{}` e `@result{}` forem recuados.

12.9.4 `@print{}` (⊣): Indicando Saída Gerada

Às vezes, uma expressão gerará saída durante a execução dela. Você pode indicar tal saída exibida com o comando `@print{}`.

O comando `@print{}` é exibido como ' ⊣ ', um traço horizontal encostado em uma barra vertical ou (quando isso não estiver disponível) a sequência ASCII '`-|`'.

No exemplo a seguir, o texto impresso é indicado com ' ⊣ ', e o valor da expressão segue na última linha.

```
(progn (print 'foo) (print 'bar))
    ⊣ foo
    ⊣ bar
    ⇒ bar
```

Em um arquivo fonte do Texinfo, esse exemplo é escrito como segue:

```
@lisp
(progn (print 'foo) (print 'bar))
    @print{} foo
    @print{} bar
    @result{} bar
@end lisp
```

12.9.5 `@error{}` (error): Indicando uma Mensagem de Erro

Um pedaço de código pode causar um erro quando você o avalia. Você pode designar a mensagem de erro com o comando `@error{}`.

O comando `@error{}` é exibido como ' error ', a palavra 'error' em uma caixa na saída impressa, a palavra error seguida por uma seta em outros formatos ou (quando nenhuma seta estiver disponível) '`error-->`'.

Assim,

```
@lisp
(+ 23 'x)
@error{} Argumento de tipo errado: integer-or-marker-p, x
@end lisp
```

produz

```
(+ 23 'x)
error  Argumento de tipo errado: integer-or-marker-p, x
```

Isso indica que a seguinte mensagem de erro é impressa quando você avalia a expressão:

```
Argumento de tipo errado: integer-or-marker-p, x
```

A própria palavra ' error ' não é parte da mensagem de erro.

12.9.6 `@equiv{}` (≡): Indicando Equivalência

Às vezes, duas expressões produzem resultados idênticos. Você pode indicar a equivalência exata de duas formas com o comando `@equiv{}`. O comando `@equiv{}` é exibido como '≡', um sinal de equivalência matemática padrão (três linhas horizontais paralelas) ou (quando isso não estiver disponível) como a sequência ASCII '`==`'.

Assim,

```
@lisp
(make-sparse-keymap) @equiv{} (list 'keymap)
@end lisp
```

produz

```
(make-sparse-keymap) ≡ (list 'keymap)
```

Isso indica que avaliar `(make-sparse-keymap)` produz resultados idênticos à avaliação de `(list 'keymap)`.

12.9.7 `@point{}` (⋆): Indicando Ponto em um Buffer

Às vezes, você precisa mostrar um exemplo de texto em um buffer do Emacs. Em tais exemplos, a convenção é a de incluir o conteúdo inteiro do buffer em questão entre duas linhas de traços contendo o nome do buffer.

Você pode usar o comando '`@point{}`' para mostrar o local do ponto no texto no buffer. (O símbolo para ponto, é claro, não é parte do texto no buffer; ele indica o lugar *entre* dois caracteres onde o ponto está localizado).

O comando `@point{}` é exibido como '⋆', uma estrela pontiaguda ou (quando essa não estiver disponível) a sequência ASCII '`-!-`'.

O exemplo a seguir mostra o conteúdo do buffer `foo` antes e depois de avaliar um comando Lisp para inserir a palavra `changed`.

```
---------- Buffer: foo ----------
Este é o ⋆conteúdo de foo.
---------- Buffer: foo ----------

(inserção "changed ")
    ⇒ nil
---------- Buffer: foo ----------
Este é o ⋆conteúdo mudado de foo.
---------- Buffer: foo ----------
```

Em um arquivo fonte do Texinfo, o exemplo é escrito assim:

```
@example
---------- Buffer: foo ----------
Este é o @point{}conteúdo de foo.
---------- Buffer: foo ----------

(inserção "changed ")
    @result{} nil
---------- Buffer: foo ----------
Este é o @point{}conteúdo mudado de foo.
---------- Buffer: foo ----------
@end example
```

12.9.8 Sequências de Clique

Ao documentar interfaces gráficas, é necessário descrever sequências como 'Clique em '`Arquivo`', depois escolha '`Abrir`', depois . . .'. O Texinfo oferece os comandos `@clicksequence` e `click` para representar isso, normalmente usados assim:

```
... @clicksequence{Arquivo @click{} Abrir} ...
```

que produz:

. . . Arquivo → Abrir . . .

O comando `@click` produz uma seta para a direita por padrão; esse glifo também está disponível independentemente por meio do comando `@arrow{}`.

Você pode mudar o glifo produzido por `@click` com o comando `@clickstyle`, que recebe um nome de comando como o único argumento dele no resto da linha, muito parecido com `@itemize` e amigos (veja Seção 9.2 [`@itemize`], Página 75). O comando deveria produzir um glifo, e as chaves vazias usuais '`{}`' são omitidas. Aqui está um exemplo:

```
@clickstyle @result
... @clicksequence{Arquivo @click{} Abrir} ...
```

agora produz:

. . . Arquivo ⇒ Abrir . . .

12.10 Inserindo Unicode: `@U`

O comando `@U{`*hex*`}` insere uma representação do caractere Unicode U+*hex*. Por exemplo, `@U{0132}` insere a ligadura holandesa 'IJ' (pobremente mostrada aqui como simplesmente as duas letras 'I' e 'J').

O valor *hex* deveria ter pelo menos quatro dígitos hexadecimais; zeros à esquerda *não* são adicionados. Em geral, *hex* precisa especificar um caractere Unicode normal válido; por exemplo, U+10FFFF (o último ponto de código) é inválido por definição e, portanto, não pode ser inserido dessa forma.

`@U` é útil para inserir glifos ocasionais para os quais Texinfo não tem comando dedicado, ao mesmo tempo em que permite que o fonte do Texinfo permaneça puramente ASCII de 7 bits para máxima portabilidade.

Esse comando tem muitas limitações—as mesmas limitações de inserir caracteres Unicode em UTF-8 ou outra forma binária. Primeiro e mais importante, TEX não sabe nada acerca da maioria do Unicode. Suportar glifos adicionais específicos mediante solicitação é possível, mas não é viável para `texinfo.tex` suportar scripts adicionais inteiros (japonês, urdu, ...). O comando `@U` não faz nada para mudar isso. Se o caractere especificado não for suportado no TEX, um erro será dado. (Veja Seção 15.2 [`@documentencoding`], Página 126).

Em HTML, XML e Docbook, a saída proveniente de `@U` é sempre uma referência de entidade do formato '`&#x`*hex*`;`', como em '`&#x0132;`' para o exemplo acima. Isso deveria funcionar mesmo quando um documento HTML usa alguma outra codificação (digamos, Latin 1) e o caractere fornecido não é suportado nessa codificação.

No Info e em texto simples, se a codificação do documento for especificada explicitamente como UTF-8, a saída será a representação UTF-8 do caractere U+*hex* (presumindo que seja um caractere válido). Em todos os outros casos, a saída é a sequência ASCII '`U+`*hex*', como nos seis caracteres ASCII '`U+0132`' para o exemplo acima.

Isso é tudo. Sem mágica!

13 Forçando e Impedindo Quebras

Quebras de linha e de página às vezes podem ocorrer no lugar 'errado' em uma ou outra forma de saída. Cabe inteiramente a você garantir que o texto aparente corretamente em todos os formatos de saída.

Por exemplo, em um manual impresso, quebras de página podem ocorrer estranhamente no meio de um exemplo; para impedir isso, você pode manter o texto unido usando um comando de agrupamento que impede que o texto seja dividido em duas páginas. Por outro lado, você pode querer forçar uma quebra de página onde nenhuma ocorreria normalmente.

Você pode usar os comandos de quebra, prevenção de quebra ou paginação para corrigir quebras de linha e de página problemáticas.

13.1 Comandos de Quebra

Os comandos de quebra criam ou permitem quebras de linha e de parágrafo:

`@*` Força uma quebra de linha.

`@sp` *n* Pula *n* linhas em branco.

`@-` Insere um hífen opcional.

`@hyphenation{`*palavras hi-fe-na-das*`}`
Define pontos de hífen em *palavras hi-fe-na-das*.

Esses comandos mantém o texto unido em uma linha:

`@w{`*texto*`}` Impede que *texto* seja dividido e hifenizado em duas linhas.

`@tie{}` Insere um espaço entre palavras normal no qual uma quebra de linha não possa ocorrer.

Os comandos de paginação aplicam-se somente para a saída impressa, uma vez que outros formatos de saída não tem páginas.

`@page` Inicia uma nova página.

`@group` Mantém unido o texto que precisa aparecer em uma página.

`@need` *mils*
Inicia uma nova página se não existir espaço suficiente nesta.

13.2 `@*` e `@/`: Gerar e Permitir Quebras de Linha

O comando `@*` força uma quebra de linha em todos os formatos de saída. O comando `@/` permite uma quebra de linha (somente manual impresso).

Aqui está um exemplo com `@*`:

```
Esta frase está dividida @*em duas linhas.
```

produz

```
Esta frase está dividida
em duas linhas.
```

O comando `@/` pode ser útil em URLs longas ou outros identificadores onde o TEX não consiga encontrar um bom lugar para quebrar. O TEX automaticamente quebrará as URLs nos lugares naturais (veja Seção 6.10.2 [Quebra de Linha de URL], Página 53), de forma que use `@/` somente se você precisar. `@/` não tem efeito no outro formato de saída.

13.3 `@-` e `@hyphenation`: Ajudando TEX Hifenizar

Embora o algoritmo de hifenização do TEX geralmente seja muito bom, ele perde pontos de hifenização úteis de vez em quando. (Ou, muito mais raramente, insere uma hifenização incorreta). Então, para documentos com um vocabulário incomum ou ao fazer ajustes finos para uma edição impressa, você possivelmente queira ajudar o TEX. Texinfo suporta dois comandos para isso:

`@-` Insere um hífen discricionário, ou seja, um lugar onde TEX pode (mas não precisa) hifenizar. Isso é especialmente útil quando você percebe que um hbox cheio demais é devido a TEX carecer de uma hifenização (veja Seção 19.10 [hboxes lotados], Página 158). TEX não inserirá ele próprio quaisquer pontos de hifenização em uma palavra contendo `@-`.

`@hyphenation{`*palavras hi-fe-na-das*`}`
Diz ao TEX como hifenizar *palavras hi-fe-na-das*. Como mostrado, você coloca um '`-`' em cada ponto de hifenização. Por exemplo:

```
@hyphenation{man-u-scrito man-u-scritos}
```

TEX só usa os pontos de hifenização especificados quando as palavras correspondem exatamente, portanto forneça todas as variantes necessárias, como plurais.

Info, HTML e outras saídas não TEX não são hifenizadas, de forma que nenhum desses comandos tem qualquer efeito ali.

13.4 `@allowcodebreaks`: Controle Quebras de Linha no `@code`

Normalmente, TEX considera a quebra de linhas nos caracteres '`-`' e '`_`' dentro do `@code` e comandos relacionados (veja Seção 7.1.2 [@code], Página 57), mais ou menos como se fossem pontos de hifenização "vazios".

Isso é necessário, pois muitos manuais, especialmente para linguagens da família Lisp, precisam documentar identificadores muito longos. Por outro lado, alguns manuais não tem esse problema, e você pode não desejar permitir uma quebra de linha no sublinhado em, por exemplo, `SIZE_MAX`, ou pior ainda, depois de qualquer um dos quatro sublinhados em `__typeof__`.

Assim, Texinfo fornece este comando:

```
@allowcodebreaks false
```

para impedir quebras em '`-`' ou '`_`' dentro de `@code`. Você pode voltar a permitir tais quebras com `@allowcodebreaks true`. Escreva esses comandos em linhas dedicadas.

Esses comandos podem ser dados em qualquer lugar no documento. Por exemplo, você pode ter apenas um parágrafo problemático onde precisa desativar as quebras, mas quer elas em geral, ou vice-versa.

Esse comando não tem efeito, exceto em saídas HTML e TEX.

13.5 `@w{`*text*`}`: Impedir Quebras de Linha

`@w{`*texto*`}` produz *texto*, ao mesmo tempo que proíbe quebras de linha dentro de *texto*.

Assim, você pode usar `@w` para produzir um espaço inquebrável, fixado na largura de um espaço normal interpalavras:

```
@w{ } @w{ } @w{ } recuo.
```

produz:

recuo.

O espaço oriundo de `@w{ }`, além de ser inquebrável, também não estica nem encolhe. Às vezes isso é o que você quer, por exemplo, se estiver fazendo recuo manual. No entanto, normalmente

você quer um espaço normal interpalavra que estica e encolhe (na saída impressa); para isso, veja o comando `@tie` na próxima seção. Você também pode usar o comando `@w` para impedir que o TEX hifenize automaticamente um nome longo ou uma frase que esteja perto do fim de uma linha. `makeinfo` nunca hifeniza palavras. Você também pode usar `@w` para evitar expansão indesejada de palavras-chave em sistemas de controle de fonte. Por exemplo, para escrever literalmente `$Id$` no teu documento, use `@w{$}Id$`. No entanto, esse truque não é eficaz no Info ou na saída de texto simples.

13.6 `@tie{}`: Inserindo um Espaço Inquebrável

O comando `@tie{}` produz um espaço normal interpalavra no qual uma quebra de linha possivelmente não ocorra. Sempre escreva-o com as seguintes chaves (vazias), como de costume para comandos usados dentro de um parágrafo. Aqui está um exemplo:

```
@TeX{} foi escrito por Donald E.@tie{}Knuth.
```

produz:

TEX foi escrito por Donald E. Knuth.

Existem duas diferenças importantes entre `@tie{}` e `@w{ }`:

- O espaço produzido por `@tie{}` será esticado e encolhido ligeiramente junto com os espaços normais entre palavras no parágrafo; o espaço produzido por `@w{ }` não variará.
- `@tie{}` permite hifenização das palavras ao redor, enquanto `@w{ }` inibe a hifenização dessas palavras (por razões TEXnológicas, ou seja, porque produz um '`\hbox`').

13.7 `@sp` *n*: Inserir Linhas em Branco

Uma linha que começa com e contém somente `@sp` *n* gera *n* linhas de espaço em branco tanto no manual impresso quanto no arquivo Info. `@sp` também força uma quebra de parágrafo. Por exemplo,

```
@sp 2
```

gera duas linhas em branco.

O comando `@sp` é usado mais frequentemente na página de título.

13.8 `@page`: Comece uma Nova Página

Uma linha contendo somente `@page` inicia uma nova página em um manual impresso. Em outros formatos, sem o conceito de páginas, ele inicia um novo parágrafo. Um comando `@page` frequentemente é usado na seção `@titlepage` de um arquivo do Texinfo para iniciar a página de direitos autorais.

13.9 `@group`: Impedir Quebras de Página

O comando `@group` (em uma linha dedicada) é usado dentro de um `@example` ou construção similar para começar um grupo vertical não divisível, que aparecerá inteiramente em uma página na saída impressa. O grupo é encerrado por uma linha contendo somente `@end group`. Essas duas linhas não produzem saída própria e, na saída do arquivo do Info, elas não tem efeito algum.

Embora `@group` faça sentido conceitualmente em uma ampla variedade de contextos, a implementação atual dele funciona confiavelmente somente dentro de `@example` e variantes, e dentro de `@display`, `@format`, `@flushleft` e `@flushright`. Veja Capítulo 8 [Citações e Exemplos], Página 66. (O que todos esses comandos tem em comum é que cada linha de entrada produz uma linha de saída). Em outros contextos, `@group` pode causar espaçamento vertical anômalo.

Essa exigência de formatação significa que você deveria escrever:

```
@example
@group
...
@end group
@end example
```

com os comandos `@group` e `@end group` dentro dos comandos `@example` e `@end example`.

O comando `@group` é mais frequentemente usado para manter um exemplo junto em uma página. Neste manual Texinfo, mais de 100 exemplos contém texto que é colocado entre `@group` e `@end group`.

Se você esquecer de encerrar um grupo, poderá receber mensagens de erro estranhas e incompreensíveis ao executar TEX. Isso ocorre porque TEX continua tentando colocar o restante do arquivo Texinfo em uma página e não começa a gerar mensagens de erro até que tenha processado texto considerável. É uma boa regra prática procurar por um `@end group` ausente se você receber mensagens de erro incompreensíveis no TEX.

13.10 `@need` *mils*: Impedir Quebras de Página

Uma linha contendo somente `@need` *n* inicia uma nova página em um manual impresso se menos que *n* mils (milésimos de polegada) permanecerem na página atual. Não use chaves ao redor do argumento *n*. O comando `@need` não tem efeito em outros formatos de saída, pois eles não são paginados.

Este parágrafo é precedido por um comando `@need` que diz ao TEX para iniciar uma nova página se restarem menos que 800 mils (oito décimos de polegada) na página. Parece algo assim:

```
@need 800
Este parágrafo é precedido por ...
```

O comando `@need` é útil para evitar órfãs: linhas únicas na parte inferior das páginas impressas.

14 Comandos de Definição

O comando `@deffn` e os outros *comandos de definição* te habilitam a descrever funções, variáveis, macros, comandos, opções de usuário(a), formulários especiais e outros tais artefatos em um formato uniforme.

No arquivo Info, uma definição faz com que a categoria da entidade—'Function', 'Variable', ou qualquer outra—apareça no início da primeira linha da definição, seguida pelo nome e argumentos da entidade. No manual impresso, o comando faz com que o TEX imprima o nome da entidade e os argumentos dela na margem esquerda e imprima a categoria próximo da margem direita. Em ambos os formatos de saída, o corpo da definição é recuado. Além disso, o nome da entidade é inserido no índice apropriado: `@deffn` insere o nome no índice de funções, `@defvr` insere-o no índice de variáveis, e assim por diante (veja Seção 11.1 [Índices Predefinidos], Página 89).

Um manual não precisa e não deveria conter mais que uma definição para um nome dado. Um anexo contendo um resumo deveria usar `@table` em vez dos comandos de definição.

14.1 O Modelo Para Uma Definição

O comando `@deffn` é usado para definições de entidades que se assemelham a funções. Para escrever uma definição usando o comando `@deffn`, escreva o comando `@deffn` no início de uma linha e siga-o na mesma linha pela categoria da entidade, o nome da entidade em si e os argumentos dela (se existirem). Em seguida, escreva o corpo da definição nas linhas seguintes. (Você pode embutir exemplos no corpo). Finalmente, termine a definição com um comando `@end deffn` escrito em uma linha própria.

Os outros comandos de definição seguem o mesmo formato: uma linha com o comando `@def...` e quaisquer argumentos apropriados para esse comando; o corpo da definição; e uma linha `@end` correspondente.

O modelo para uma definição se parece com isto:

```
@deffn categoria nome argumentos...
corpo-da-definição
@end deffn
```

Por exemplo,

```
@deffn Command forward-word count
Esse comando move o ponto para frente @var{count} palavras (ou para trás, se
@var{count} for negativo). ...
@end deffn
```

produz

`forward-word` *count* [Command]
: Esse comando move o ponto para frente *count* palavras (ou para trás, se *count* for negativo). ...

Coloque o nome da categoria em maiúsculas como um título. Se o nome da categoria contiver espaços, como na frase 'Comando Interativo', coloque-o entre chaves. Por exemplo:

```
@deffn {Comando Interativo} isearch-forward
...
@end deffn
```

Caso contrário, a segunda palavra será confundida com o nome da entidade. Como regra geral, quando quaisquer dos argumentos na linha de título, *exceto* o último, forem mais que uma palavra, você precisa colocá-los entre chaves. Isso também pode ser necessário se o texto contiver comandos, por exemplo, '`{declaraci@'on}`' se você estiver escrevendo em espanhol.

Alguns dos comandos de definição são mais gerais que outros. O comando `@deffn`, por exemplo, é o comando geral de definição para funções e similares—para entidades que possivelmente recebam argumentos. Quando usa esse comando, você especifica a categoria à qual a entidade pertence. Três variações predefinidas e especializadas (`@defun`, `@defmac` e `@defspec`) especificam a categoria para você: "Função", "Macro" e "Forma Especial" respectivamente. (Em Lisp, uma forma especial é uma entidade muito parecida com uma função). Similarmente, o comando geral `@defvr` é acompanhado por diversas variações especializadas para descrever tipos particulares de variáveis.

Veja Seção 14.7 [Definição de Função de Amostra], Página 123, para um exemplo detalhado de uma definição de função, incluindo o uso de `@example` dentro da definição.

14.2 Linhas de Continuação de Comando de Definição

A linha de título de um comando de definição pode ficar muito longa. Portanto, o Texinfo tem uma sintaxe especial que permite que elas sejam continuadas em várias linhas do arquivo fonte: um solitário '`@`' no final de cada linha a ser continuada. Aqui está um exemplo:

```
@defun fn-name @
  arg1 arg2 arg3
Essa é a defun básica continuada.
@end defun
```

produz:

`fn-name` *arg1 arg2 arg3* [Função]

Essa é a defun básica continuada.

Como você pode ver, as linhas contínuas estão combinadas, como se tivessem sido digitadas em uma linha do fonte.

Embora esse exemplo mostre somente uma continuação de uma linha, as continuações podem se estender por qualquer número de linhas, da mesma forma; coloque um `@` no final de cada linha a ser continuada.

Em geral, qualquer número de espaços ou tabulações antes do caractere de continuação `@` é recolhido em um espaço. Existe uma exceção: os processadores Texinfo não recolherão totalmente espaços em branco em torno de uma continuação dentro de chaves. Por exemplo:

```
@deffn {Nome @
  Categoria} ...
```

A saída gerada (não mostrada) tem excesso de espaço entre 'Nome' e 'Categoria'. Para evitar isso, elimine os espaços em branco indesejados na tua entrada ou coloque a continuação `@` fora das chaves.

`@` não funciona como um caractere de continuação em *qualquer* outro contexto. Normalmente, '`@`' seguido por um caractere de espaço em branco (espaço, tabulação, nova linha) produz um espaço normal interpalavra (veja Seção 12.3.1 [Espaços Múltiplos], Página 97).

14.3 Argumentos Opcionais e Repetidos

Algumas entidades aceitam argumentos opcionais ou repetidos, convencionalmente especificados usando-se colchetes e reticências: um argumento entre colchetes é opcional, e um argumento seguido por reticências é opcional e pode ser repetido mais que uma vez.

Assim, [*argumento-opcional*] significa que *argumento-opcional* é opcional e *argumentos-repetidos*... representa zero ou mais argumentos. Parênteses são usados quando vários argumentos são agrupados em níveis adicionais de estrutura de lista na Lisp.

Aqui está a linha `@defspec` de um exemplo de uma forma especial (complicada) imaginária:

`foobar` (*var* [*from to* [*inc*]]) *body*... [Forma Especial]

Nesse exemplo, os argumentos *from* e *to* são opcionais, mas ambos precisam estar presentes ou ambos ausentes. Se estiverem presentes, *inc* opcionalmente pode ser especificado também. Esses argumentos são agrupados com o argumento *var* em uma lista, para distingui-los de *body*, que inclui todos os elementos restantes da forma.

Em um arquivo fonte do Texinfo, essa linha `@defspec` é escrita assim:

```
@defspec foobar (var [from to [inc]]) body@dots{}
```

A função está listada no Índice de Comando e Variável sob '`foobar`'.

14.4 `@deffnx`, et al.: Duas ou Mais 'Primeiras' Linhas

Para criar duas ou mais linhas 'primeira' ou de cabeçalho para uma definição, siga a primeira linha `@deffn` por uma linha começando com `@deffnx`. O comando `@deffnx` funciona exatamente como `@deffn`, exceto que ele não gera espaço em branco vertical extra entre ele e a linha precedente.

Por exemplo,

```
@deffn {Comando Interativo} isearch-forward
@deffnx {Comando Interativo} isearch-backward
Esses dois comandos de pesquisa são semelhantes, exceto ...
@end deffn
```

produz

`isearch-forward` [Comando Interativo]
`isearch-backward` [Comando Interativo]

Esses dois comandos de pesquisa são semelhantes, exceto ...

Cada comando de definição tem uma forma 'x': `@defunx`, `@defvrx`, `@deftypefunx`, etc.

As formas 'x' funcionam similarmente a `@itemx` (veja Seção 9.4.3 [`@itemx`], Página 79).

14.5 Os Comandos de Definição

Texinfo fornece mais que uma dúzia de comandos de definição, todos estão descritos nesta seção.

Os comandos de definição inserem automaticamente o nome da entidade no índice apropriado: por exemplo, `@deffn`, `@defun` e `@defmac` inserem nomes de funções no índice de funções; `@defvr` e `@defvar` inserem nomes de variáveis no índice de variáveis.

Embora os exemplos a seguir ilustrem principalmente Lisp, os comandos podem ser usados para outras linguagens de programação.

14.5.1 Funções e Entidades Similares

Esta seção descreve os comandos para descrever funções e entidades similares:

`@deffn` *categoria nome argumentos...*

O comando `@deffn` é o comando geral de definição para funções, comandos interativos e entidades similares que podem receber argumentos. Você precisa escolher um termo para descrever a categoria da entidade sendo definida; por exemplo, "Function" poderia ser usado se a entidade for uma função. O comando `@deffn` é escrito no início de uma linha e é seguido na mesma linha pela categoria da entidade sendo descrita, o nome dessa entidade em particular e os argumentos dela, se existirem. Termine a definição com `@end deffn` em uma linha própria.

Por exemplo, aqui está uma definição:

```
@deffn Comando forward-char nchars
Move ponto para frente @var{nchars} caracteres.
@end deffn
```

Isso mostra uma definição bastante concisa para um "comando" chamado `forward-char` com um argumento, *nchars*.

`@deffn` imprime nomes de argumentos como *nchars* em tipo inclinado na saída impressa, porque nós pensamos nesses nomes como variáveis metassintáticas—elas representam os valores reais dos argumentos. No texto da descrição, no entanto, escreva um nome de argumento explicitamente com `@var` para se referir ao valor do argumento. No exemplo acima, nós usamos '`@var{nchars}`' dessa forma.

No caso extremamente incomum quando um nome de argumento contiver '`--`', ou outra sequência de caracteres que seja tratada especialmente (veja Seção 2.1 [Convenções], Página 9), use `@code` ao redor dos caracteres especiais. Isso evita a conversão para traços de ligação e travessões tipográficos.

O modelo para `@deffn` é:

```
@deffn categoria nome argumentos...
corpo-da-definição
@end deffn
```

`@defun` *nome argumentos...*

O comando `@defun` é o comando de definição para funções. `@defun` é equivalente a '`@deffn Função ...`'. Termine a definição com `@end defun` em uma linha própria. Assim, o modelo é:

```
@defun nome-função argumentos...
corpo-da-definição
@end defun
```

`@defmac` *nome argumentos...*

O comando `@defmac` é o comando de definição para macros. `@defmac` é equivalente a '`@deffn Macro ...`' e funciona como `@defun`.

`@defspec` *nome argumentos...*

O comando `@defspec` é o comando de definição para formas especiais. (Na Lisp, uma forma especial é uma entidade muito parecida com uma função; veja Seção "Special Forms" em *GNU Emacs Lisp Reference Manual*). `@defspec` é equivalente a '`@deffn {Forma Especial} ...`' e funciona como `@defun`.

Todos esses comandos criam entradas no índice de funções.

14.5.2 Variáveis e Entidades Similares

Aqui estão os comandos para definir variáveis e entidades similares:

`@defvr` *categoria nome*

O comando `@defvr` é um comando geral de definição para algo como uma variável—uma entidade que registra um valor. Você precisa escolher um termo para descrever a categoria da entidade sendo definida; por exemplo, "Variável" poderia ser usado se a entidade for uma variável. Escreva o comando `@defvr` no início de uma linha e siga-o na mesma linha pela categoria da entidade e o nome da entidade.

Nós recomendamos colocar o nome da categoria em maiúsculas como um título. Se o nome da categoria contiver espaços, como no nome "Opção de Usuário(a)", coloque-o entre chaves. Caso contrário, a segunda palavra será confundida com o nome da entidade. Por exemplo,

```
@defvr {Opção de Usuário(a)} fill-column
Essa variável local e de buffer especifica a largura máxima das linhas preen
...
@end defvr
```

Termine a definição com `@end defvr` em uma linha própria.

O modelo é:

```
@defvr categoria nome
corpo-da-definição
@end defvr
```

`@defvr` cria uma entrada no índice de variáveis para *nome*.

`@defvar` *nome*

O comando `@defvar` é o comando de definição para variáveis. `@defvar` é equivalente a '`@defvr Variável ...`'.

Por exemplo:

```
@defvar kill-ring
...
@end defvar
```

O modelo é:

```
@defvar nome
corpo-da-definição
@end defvar
```

`@defvar` cria uma entrada no índice de variáveis para *nome*.

`@defopt` *nome*

O comando `@defopt` é o comando de definição para *opções de usuário(a)*, ou seja, variáveis destinadas para usuários(as) mudarem de acordo com o gosto; Emacs tem muitas dessas (veja Seção "Variables" em *O Manual do GNU Emacs*). `@defopt` é equivalente a '`@defvr {Opção de Usuário(a)} ...`' e funciona como `@defvar`. Ele cria uma entrada no índice de variáveis.

14.5.3 Funções em Linguagens Tipadas

O comando `@deftypefn` e as variações dele são para descrever funções em linguagens nas quais você precisa declarar tipos de variáveis e funções, como C e C++.

`@deftypefn` *categoria tipo-dados nome argumentos...*

O comando `@deftypefn` é o comando geral de definição para funções e entidades similares que podem receber argumentos e que são tipadas. O comando `@deftypefn` é escrito no início de uma linha e é seguido na mesma linha pela categoria da entidade sendo descrita, o tipo do valor retornado, o nome dessa entidade em particular e os argumentos dela, se existirem.

Por exemplo,

```
@deftypefn {Função de Biblioteca} int foobar @
  (int @var{foo}, float @var{bar})
...
@end deftypefn
```

produz:

`int foobar` (*int foo*, *float bar*) [Função de Biblioteca]

...

Isso significa que `foobar` é uma "função de biblioteca" que retorna um `int`, e os argumentos dela são *foo* (um `int`) e *bar* (um `float`).

Como em linguagens tipadas, os nomes reais dos argumentos são normalmente espalhados entre nomes de tipos de dados e palavras-chave, Texinfo não consegue

encontrá-los sem ajuda. Você pode (a) escrever tudo como texto simples, e ele será impresso em tipo inclinado; (b) usar `@var` para os nomes de variáveis, que colocará os nomes de variáveis em maiúsculas no Info e usará a fonte de máquina de escrever inclinada na saída impressa; (c) usar `@var` para os nomes de variáveis e `@code` para os nomes de tipos e palavras-chave, que serão obedientemente obedecidos.

O modelo para `@deftypefn` é:

```
@deftypefn categoria tipo-dados nome argumentos ...
corpo-da-descrição
@end deftypefn
```

Observe que se *categoria* ou *tipo-dados* for mais de uma palavra, então ele precisa ser colocado entre chaves para torná-lo um argumento unitário.

Se você estiver descrevendo um procedimento em uma linguagem que tenha pacotes, como Ada, você pode considerar usar `@deftypefn` de uma maneira um tanto contrária à convenção descrita nos parágrafos precedentes. Por exemplo:

```
@deftypefn stacks private push @
       (@var{s}:in out stack; @
       @var{n}:in integer)
...
@end deftypefn
```

(Nesses exemplos os argumentos `@deftypefn` são mostrados usando continuações (veja Seção 14.2 [Linhas de Continuação de Comando de Definição], Página 115), mas poderiam estar em uma linha).

Nesse caso, o procedimento é classificado como pertencente ao pacote `stacks` em vez de classificado como um 'procedimento' e o tipo de dados dele está descrito como `private`. (O nome do procedimento é `push`, e os argumentos dele são *s* e *n*).

`@deftypefn` cria uma entrada no índice de funções para *nome*.

`@deftypefun` *tipo-dados nome argumentos...*

O comando `@deftypefun` é o comando especializado de definição para funções em linguagens tipadas. O comando é equivalente a '`@deftypefn Função ...`'. O modelo é:

```
@deftypefun tipo nome argumentos...
corpo-da-descrição
@end deftypefun
```

`@deftypefun` cria uma entrada no índice de funções para *nome*.

Normalmente, o tipo de retorno é impresso na mesma linha que o nome da função e os argumentos, como mostrado acima. No código-fonte, o estilo GNU é o de colocar o tipo de retorno em uma linha dedicada. Portanto, Texinfo fornece uma opção para fazer isso: `@deftypefnnewline on`.

Isso afeta somente funções tipadas—não funções não tipadas, variáveis não tipadas, etc.. Especificamente, afeta os comandos nesta seção e os comandos análogos para linguagens orientadas a objetos, ou seja, `@deftypeop` e `@deftypemethod` (veja Seção 14.5.6.2 [Métodos Orientados a Objetos], Página 122).

Especificar-se `@deftypefnnewline off` reverte para o padrão.

14.5.4 Variáveis em Linguagens Tipadas

Variáveis em linguagens tipadas são manuseadas de forma similar a funções em linguagens tipadas. Veja Seção 14.5.3 [Funções Tipadas], Página 118. O comando geral de definição

`@deftypevr` corresponde a `@deftypefn` e o comando especializado de definição `@deftypevar` corresponde a `@deftypefun`.

`@deftypevr` *categoria tipo-dados nome*

O comando `@deftypevr` é o comando geral de definição para algo como uma variável em uma linguagem tipada—uma entidade que registra um valor. Você precisa escolher um termo para descrever a categoria da entidade sendo definida; por exemplo, "Variável" poderia ser usado se a entidade for uma variável.

O comando `@deftypevr` é escrito no início de uma linha e é seguido na mesma linha pela categoria da entidade sendo descrita, o tipo de dados e o nome dessa entidade específica.

Por exemplo:

```
@deftypevr {Sinalizador Global} int enable
...
@end deftypevr
```

produz o seguinte:

`int enable` [Sinalizador Global]
...

O modelo é:

```
@deftypevr categoria tipo-dados nome
corpo-da-descrição
@end deftypevr
```

`@deftypevar` *tipo-dados nome*

O comando `@deftypevar` é o comando especializado de definição para variáveis em linguagens tipadas. `@deftypevar` é equivalente a '`@deftypevr Variável ...`'. O modelo é:

```
@deftypevar tipo-dados nome
corpo-da-descrição
@end deftypevar
```

Esses comandos criam entradas no índice de variáveis.

14.5.5 Tipos de Dados

Aqui está o comando para tipos de dados:

`@deftp` *categoria nome atributos...*

O comando `@deftp` é o comando genérico de definição para tipos de dados. O comando é escrito no início de uma linha e é seguido na mesma linha pela categoria, pelo nome do tipo (que é uma palavra como `int` ou `float`) e, em seguida, pelos nomes de atributos de objetos desse tipo. Portanto, você poderia usar esse comando para descrever `int` ou `float`, em cujo caso você poderia usar `tipo de dados` como a categoria. (Um tipo de dados é uma categoria de certos objetos para fins de decidir quais operações podem ser realizadas sobre neles).

Na Lisp, por exemplo, *pair* nomeia um tipo de dado particular, e um objeto desse tipo tem dois slots chamados CAR e CDR. Aqui está como você escreveria a primeira linha de uma definição de `pair`.

```
@deftp {Tipo de dados} pair car cdr
...
@end deftp
```

O modelo é:

```
@deftp categoria nome-do-tipo atributos...
corpo-da-definição
@end deftp
```

`@deftp` cria uma entrada no índice de tipos de dados.

14.5.6 Programação Orientada a Objetos

Aqui estão os comandos para formatar descrições acerca de objetos abstratos, tais como são usados em programação orientada a objetos. Uma classe é um tipo definido de objeto abstrato. Uma instância de uma classe é um objeto particular que tem o tipo da classe. Uma variável de instância é uma variável que pertence à classe, mas para a qual cada instância tem o próprio valor dela.

14.5.6.1 Variáveis Orientadas a Objetos

Estes comandos permitem que você defina diferentes tipos de variáveis em linguagens de programação orientadas a objetos.

`@defcv` *categoria classe nome*

O comando `@defcv` é o comando geral de definição para variáveis associadas a classes em programação orientada a objetos. O comando `@defcv` é seguido por três argumentos: a categoria da coisa sendo definida, a classe à qual ela pertence e o nome dela. Por exemplo:

```
@defcv {Opção da Classe} Janela padrão-borda
...
@end defcv
```

produz:

`padrão-borda` [Opção da Classe de `Janela`]
...

`@defcv` cria uma entrada no índice de variáveis.

`@deftypecv` *categoria classe tipo-dados nome*

O comando `@deftypecv` é o comando de definição para variáveis de classe tipadas em programação orientada a objetos. Ele é análogo ao `@defcv` com a adição do parâmetro *tipo-dados* para especificar o tipo da variável de instância. Normalmente, o tipo de dado é uma construção da linguagem de programação que deveria estar marcada com `@code`. Por exemplo:

```
@deftypecv {Opção da Classe} Janela @code{int} padrão-borda
...
@end deftypecv
```

produz:

`int padrão-borda` [Opção da Classe de `Janela`]
...

`@deftypecv` cria uma entrada no índice de variáveis.

`@defivar` *classe nome*

O comando `@defivar` é o comando de definição para variáveis de instância em programação orientada a objetos. `@defivar` é equivalente a '`@defcv {Variável da Instância} ...`'. Por exemplo:

```
@defivar Janela padrão-borda
...
@end defivar
```

produz:

`padrão-borda` [Instance Variable of `Janela`]
...

`@defivar` cria uma entrada no índice de variáveis.

`@deftypeivar` *classe tipo-dados nome*
O comando `@deftypeivar` é o comando de definição para variáveis de instância tipadas em programação orientada a objetos. Ele é análogo ao `@defivar` com a adição do parâmetro *tipo-dados* para especificar o tipo da variável de instância. Normalmente, o tipo de dado é uma construção da linguagem de programação que deveria estar marcada com `@code`. Por exemplo:

```
@deftypeivar Janela @code{int} padrão-borda
...
@end deftypeivar
```

produz:

`int padrão-borda` [Instance Variable of `Janela`]
...

`@deftypeivar` cria uma entrada no índice de variáveis.

14.5.6.2 Métodos Orientados a Objetos

Esses comandos permitem que você defina diferentes tipos de entidades semelhantes a funções que lembram métodos em linguagens de programação orientadas a objetos. Essas entidades recebem argumentos, como as funções, mas são associadas a classes particulares de objetos.

`@defop` *categoria classe nome argumentos...*
O comando `@defop` é o comando geral de definição para essas entidades semelhantes a métodos.

Por exemplo, alguns sistemas tem construções chamadas *involucradores* que são associadas a classes como métodos, mas que agem mais como macros que como funções. Você poderia usar `@defop Wrapper` para descrever uma delas.

Ocasionalmente, é útil distinguir métodos e *operações*. Você pode pensar em uma operação como a especificação para um método. Assim, um sistema de janelas pode especificar que todas as classes de janelas tem um método chamado `expor`; nós diríamos que esse sistema de janelas define uma operação `expor` sobre janelas em geral. Normalmente, a operação tem um nome e também especifica o padrão de argumentos; todos os métodos que implementem a operação precisam aceitar os mesmos argumentos, já que os aplicativos que usam a operação o fazem sem saber qual método a implementará.

Frequentemente faz mais sentido documentar operações que métodos. Por exemplo, desenvolvedores(as) de aplicativos de janela precisam saber acerca da operação `expor`, mas não precisam se preocupar se uma determinada classe de janelas tem o próprio método dela para implementar essa operação. Para descrever essa operação, você escreveria:

```
@defop Operação janelas expor
```

O comando `@defop` é escrito no início de uma linha e é seguido na mesma linha pelo nome geral da categoria da operação, pelo nome da classe da operação, pelo nome da operação e pelos argumentos dela, se existirem.

O modelo é:

```
@defop categoria classe nome argumentos...
corpo-da-definição
@end defop
```

`@defop` cria uma entrada, como '`expor` sobre `janelas`', no índice de funções.

`@deftypeop` *categoria classe tipo-dados nome argumentos...*

O comando `@deftypeop` é o comando de definição para operações tipadas em programação orientada a objetos. Ele é semelhante ao `@defop` com a adição do parâmetro *tipo-dados* para especificar o tipo de retorno do método. `@deftypeop` cria uma entrada no índice de funções.

`@defmethod` *classe nome argumentos...*

O comando `@defmethod` é o comando de definição para métodos em programação orientada a objetos. Um método é um tipo de função que implementa uma operação para uma classe particular de objetos e as subclasses dela.

`@defmethod` é equivalente a '`@defop Método ...`'. O comando é escrito no início de uma linha e é seguido pelo nome da classe do método, pelo nome do método e pelos argumentos dele, se existirem.

Por exemplo:

```
@defmethod classe-barra método-barra argumento
...
@end defmethod
```

ilustra a definição para um método chamado `método-barra` da classe `classe-barra`. O método recebe um argumento.

`@defmethod` cria uma entrada no índice de funções.

`@deftypemethod` *classe tipo-dados nome argumentos...*

O comando `@deftypemethod` é o comando de definição para métodos em linguagens tipadas e orientadas a objetos, como C++ e Java. Ele é semelhante ao comando `@defmethod` com a adição do parâmetro *tipo-dados* para especificar o tipo de retorno do método. `@deftypemethod` cria uma entrada no índice de funções.

Os comandos tipados são afetados pela opção `@deftypefnnewline` (veja Seção 14.5.3 [Funções em Linguagens Tipadas], Página 118).

14.6 Convenções para Escrita de Definições

Quando você escrever uma definição usando `@deffn`, `@defun` ou um dos outros comandos de definição, por favor, tome cuidado para usar argumentos que indiquem o significado, como com o argumento *count* para a função `forward-word`. Além disso, se o nome de um argumento contiver o nome de um tipo, como *integer*, tome cuidado para que o argumento seja realmente desse tipo.

14.7 Uma Definição de Função de Amostra

Uma definição de função usa os comandos `@defun` e `@end defun`. O nome da função vem imediatamente depois do comando `@defun` e é seguido, na mesma linha, pela lista de parâmetros.

Aqui está uma definição oriunda de Seção "Calling Functions" em *The GNU Emacs Lisp Reference Manual*.

`apply` *function* **&rest** *arguments* [Função]

`apply` chama *function* com *arguments*, assim como `funcall`, mas com uma diferença: o último de *arguments* é uma lista de argumentos para fornecer para

function, em vez de um argumento unitário. Nós também dizemos que essa lista é *anexada* aos outros argumentos.

`apply` retorna o resultado da chamada de *function*. Assim como com `funcall`, *function* precisa ser ou uma função da Lisp ou uma função primitiva; formas especiais e macros não fazem sentido em `apply`.

```
(setq f 'list)
    ⇒ list
(apply f 'x 'y 'z)
error  Argumento de tipo errado: listp, z
(apply '+ 1 2 '(3 4))
    ⇒ 10
(apply '+ '(1 2 3 4))
    ⇒ 10

(apply 'append '((a b c) nil (x y z) nil))
    ⇒ (a b c x y z)
```

Um exemplo interessante do uso de `apply` é encontrado na descrição de `mapcar`.

No arquivo fonte do Texinfo, esse exemplo se parece com isto:

```
@defun apply function &rest arguments
@code{apply} chama @var{function} com @var{arguments}, assim como
@code{funcall}, mas com uma diferença: o último de @var{arguments} é uma
lista de argumentos para fornecer para @var{function}, em vez de um
argumento unitário. Nós também dizemos que essa lista é @dfn{anexada} aos
outros argumentos.

@code{apply} retorna o resultado da chamada de @var{function}. Assim como
com @code{funcall}, @var{function} precisa ser ou uma função da Lisp ou
uma função primitiva; formas especiais e macros não fazem sentido em
@code{apply}.

@example
(setq f 'list)
    @result{} list
(apply f 'x 'y 'z)
@error{} Argumento de tipo errado: listp, z
(apply '+ 1 2 '(3 4))
    @result{} 10
(apply '+ '(1 2 3 4))
    @result{} 10

(apply 'append '((a b c) nil (x y z) nil))
    @result{} (a b c x y z)
@end example

Um exemplo interessante do uso de @code{apply} é encontrado na descrição de
@code{mapcar}.
@end defun
```

Neste manual, essa função está listada no Índice de Comandos e Variáveis sob `apply`.

Variáveis comuns e opções de usuário(a) são descritas usando um formato semelhante àquele para funções, exceto que variáveis não recebem argumentos.

15 Internacionalização

Texinfo tem algum suporte para escrita em outros idiomas além do inglês, embora essa área ainda precise de trabalho considerável. (Se você mesmo(a) estiver ajudando a traduzir as strings fixas escritas em documentos, veja Seção 20.6 [Internacionalização de Strings de Documentos], Página 182).

Para uma lista dos vários caracteres acentuados e especiais que Texinfo suporta, veja-se Seção 12.4 [Inserindo Acentos], Página 100.

15.1 `@documentlanguage` *ll*`[_`*CC*`]`: Configurar Idioma do Documento

O comando `@documentlanguage` declara a localidade atual do documento. Escreva-o em uma linha dedicada, perto do começo do arquivo.

```
@documentlanguage ll[_CC]
```

Inclua um código de idioma ISO 639-2 de duas letras (*ll*) seguindo o nome do comando, opcionalmente seguido por um sublinhado e um código de país ISO 3166 de duas letras (*CC*). Se você tiver um documento multi idioma, a intenção é a de poder usar esse comando várias vezes para declarar cada mudança de idioma. Se o comando não for usado, o padrão é `en_US` para inglês dos Estados Unidos da América do Norte.

Assim como com GNU Gettext (veja *Gettext*), se o código do país for omitido, o dialeto principal é assumido onde possível. Por exemplo, `de` é equivalente a `de_DE` (alemão como falado na Alemanha).

Para Info e outras saídas on-line, esse comando muda a tradução de várias *strings de documento*, tais como "see" em referências cruzadas (veja Capítulo 6 [Referências Cruzadas], Página 45), "Function" em defuns (veja Capítulo 14 [Comandos de Definição], Página 114), e assim por diante. Algumas strings, como "Node:", "Next:", "Menu:", etc., são palavras-chave na saída Info, de forma que não são traduzidas lá; elas são traduzidas em outros formatos de saída.

Para TEX, esse comando faz com que um arquivo `txi-`*locale*`.tex` seja lido (se ele existir). Se o argumento `@documentlanguage` contiver o sufixo opcional '`_cc`', isso será tentado primeiro. Por exemplo, com `@documentlanguage de_DE`, TEX primeiro procura por `txi-de_DE.tex`, depois por `txi-de.tex`.

Esse arquivo `txi-*` é destinado a redefinir as várias palavras em inglês usadas na saída do TEX, como 'Chapter', 'See' e assim por diante. Nós estamos cientes de que palavras individuais como essas nem sempre podem ser traduzidas isoladamente, e que uma estratégia muito diferente seria exigida para scripts ideográficos (entre outros). Ajuda para melhorar o suporte a idiomas do Texinfo é bem-vinda.

`@documentlanguage` também muda os padrões de hifenização atuais do TEX, se o programa TEX sendo executado tiver o suporte necessário incluído. Isso geralmente não será o caso para `tex` em si, mas geralmente será o caso para distribuições atualizadas dos programas TEX estendidos `etex` (saída DVI) e `pdftex` (saída PDF). `texi2dvi` usará os TEXs estendidos se estiverem disponíveis (veja Seção 19.2 [Formatar com `texi2dvi`], Página 150).

Em setembro de 2006, a W3C Internationalization Activity lançou uma nova recomendação para especificar idiomas: `http://www.rfc-editor.org/rfc/bcp/bcp47.txt`. Quando Gettext suportar esse novo esquema, Texinfo também suportará.

Como as listas de códigos de idioma e códigos de país são atualizadas com relativa frequência, nós não tentamos listá-los aqui. Os códigos válidos de idioma estão na página inicial oficial do ISO 639, `http://www.loc.gov/standards/iso639-2/`. Os códigos de país e o site oficial do ISO 3166 podem ser encontrados via `http://en.wikipedia.org/wiki/ISO_3166`.

15.2 `@documentencoding` *enc*: Configurar Codificação de Entrada

O comando `@documentencoding` declara a codificação de entrada do documento e também pode afetar a codificação da saída. Escreva-o em uma linha dedicada, com uma especificação válida de codificação seguindo, perto do início do arquivo.

```
@documentencoding codificação
```

Texinfo suporta estas codificações:

`US-ASCII` Isso não tem nenhum efeito específico, mas foi incluído para fins de completude.

`UTF-8` A vasta codificação global de caracteres, expressa em bytes de 8 bits.

`ISO-8859-1`
`ISO-8859-15`
`ISO-8859-2`
: Essas especificam as codificações padrão para idiomas da Europa Ocidental (os dois primeiros) e da Europa Oriental (o terceiro), respectivamente. A ISO 8859-15 substitui alguns caracteres pouco usados provenientes da 8859-1 (por exemplo, frações pré-compostas) por outros mais comumente necessários, como o símbolo do Euro ().

  Uma descrição completa das codificações está além do nosso escopo aqui; uma referência útil é `http://czyborra.com/charsets/iso8859.html`.

`koi8-r` Essa é a codificação comumente usada para o idioma russo.

`koi8-u` Essa é a codificação comumente usada para o idioma ucraniano.

Especificar uma codificação *codificação* tem os seguintes efeitos:

Na saída do Info, uma seção chamada 'Local Variables' (veja Seção "File Variables" em *O Manual do GNU Emacs*) é gerada incluindo *codificação*. Isso permite que leitores Info configurem a codificação apropriadamente. Parece com algo assim:

```
Local Variables:
coding: codificação
End:
```

Além disso, na saída Info e na de texto simples, a menos que a opção `--disable-encoding` seja fornecida para `makeinfo`, construções de acentos e caracteres especiais, como `@'e`, são produzidos como o caractere real de 8 bits ou UTF-8 na codificação fornecida, onde possível.

Na saída HTML, uma etiqueta '`<meta>`' é gerada, na seção '`<head>`' do HTML, que especifica *codificação*. Servidores web e navegadores cooperam para usar essas informações, de modo que a codificação correta seja usada para exibir a página, se suportada pelo sistema. Isso se parece com isto:

```
<meta http-equiv="Content-Type" content="text/html;
     charset=codificação">
```

Na saída XML e Docbook, UTF-8 é sempre usado para a saída, de acordo com as convenções desses formatos.

Na saída TEX, os caracteres que são suportados nas fontes Computer Modern padrão são produzidos adequadamente. Por exemplo, isso significa usar acentos construídos em vez de glifos pré-compostos. Usar um caractere ausente gera uma mensagem de aviso, assim como especificar uma codificação não implementada.

Embora os sistemas TEX modernos suportem quase todos os scripts em uso no mundo, esse suporte abrangente não está disponível em `texinfo.tex`, e não é viável duplicar ou incorporar todo esse esforço. (Nosso plano para suportar outros scripts é o de criar uma estrutura de retaguarda LATEX para `texi2any`, onde o suporte já está presente).

Para portabilidade máxima de documentos Texinfo entre os muitos ambientes de usuário(a) diferentes no mundo, nós recomendamos manter ASCII de 7 bits na entrada, a menos que teu

manual em particular precise de uma quantidade substancial de não-ASCII, por exemplo, ele esteja escrito em alemão. Você pode usar o comando `@U` para inserir um caractere ocasionalmente necessário (veja Seção 12.10 [Inserindo Unicode], Página 108).

16 Texto Visível Condicionalmente

Os *comandos condicionais* permitem que você use texto diferente para formatos de saída diferentes, ou para condições gerais que você define. Por exemplo, você pode usá-los para especificar texto diferente para o manual impresso e para a saída Info.

Os comandos condicionais compreendem as seguintes categorias.

- Comandos específicos para um formato de saída (Info, TEX, HTML, ...).
- Comandos específicos para qualquer formato de saída *excluindo* um determinado (por exemplo, não Info, não TEX, ...).
- Texto do formatador 'bruto' para qualquer formato de saída, passado diretamente com interpretação mínima (mas não nula) dos comandos @.
- Substituições de variáveis independentes de formato e testagem se uma variável está configurada ou limpa.

16.1 Comandos Condicionais

O Texinfo tem um ambiente `@if`*`format`* para cada formato de saída, para permitir inclusão condicional de texto para um formato de saída específico.

`@ifinfo` inicia segmentos de texto que deveriam ser ignorados pelo TEX quando ele compuser o manual impresso, e pelo `makeinfo` quando não produzir saída Info. O segmento de texto aparece somente no arquivo Info e, para compatibilidade histórica, na saída de texto simples.

Os ambientes para os outros formatos são análogos:

`@ifdocbook ... @end ifdocbook`
: Texto para aparecer somente na saída Docbook.

`@ifhtml ... @end ifhtml`
: Texto para aparecer somente na saída HTML.

`@ifplaintext ... @end ifplaintext`
: Texto para aparecer somente na saída de texto simples.

`@iftex ... @end iftex`
: Texto para aparecer somente no manual impresso.

`@ifxml ... @end ifxml`
: Texto para aparecer somente na saída XML.

Os comandos `@if...` e `@end if...` precisam aparecer em linhas dedicadas no teu arquivo fonte. As quebras de linha que seguem os comandos são (mais ou menos) tratadas como espaços em branco, de forma que o texto condicional flua normalmente para um parágrafo circundante.

As construções `@if...` são destinadas para condicionar fonte Texinfo normal; veja Seção 16.3 [Comandos do Formatador Bruto], Página 130, para usar comandos de formatação subjacentes diretamente.

Aqui está um exemplo mostrando todos esses Condicionais:

```
@iftex
Este texto aparecerá somente no manual impresso.
@end iftex
@ifinfo
No entanto, este texto aparecerá somente em Info e em texto simples.
@end ifinfo
@ifhtml
E este texto aparecerá somente em HTML.
```

```
@end ifhtml
@ifplaintext
Enquanto este texto aparecerá somente em texto simples.
@end ifplaintext
@ifxml
Não obstante, isto aparecerá somente em XML.
@end ifxml
@ifdocbook
No entanto, isto aparecerá somente em Docbook.
@end ifdocbook
```

O exemplo precedente produz a seguinte linha:

Este texto aparecerá somente no manual impresso.

Observe que você vê somente uma das linhas de entrada, dependendo de qual versão do manual estiver lendo.

Em documentos complexos, você pode querer que o Texinfo emita uma mensagem de erro em alguns Condicionais que nunca deveriam ser processados. O comando `@errormsg{text}` fará isso; ele pega um argumento, o texto da mensagem de erro, que é expandido mais ou menos como se fosse um texto do Info.

Nós mencionamos `@errormsg{}` aqui, embora não esteja estritamente relacionado a Condicionais, já que, na prática, é mais provável que seja útil nesse contexto. Tecnicamente, ele pode ser usado em qualquer lugar. Veja Seção 17.6 [Processadores Externos de Macro], Página 143, para uma ressalva relativa aos números de linha que `@errormsg` emite no TEX.

16.2 Não Comandos Condicionais

Você pode especificar o texto a ser incluído em qualquer formato de saída *diferente* de um determinado, com os ambientes `@ifnot...`:

```
@ifnotdocbook ... @end ifnotdocbook
@ifnothtml ... @end ifnothtml
@ifnotinfo ... @end ifnotinfo
@ifnotplaintext ... @end ifnotplaintext
@ifnottex ... @end ifnottex
@ifnotxml ... @end ifnotxml
```

O comando `@ifnot...` e o comando `@end` precisam aparecer em linhas dedicadas no teu arquivo fonte real.

Se o arquivo de saída estiver sendo feito no formato fornecido, a região será *ignorada*. Caso contrário, ela será incluída.

Existe uma exceção (para compatibilidade histórica): texto `@ifnotinfo` será omitido para ambas saídas, Info e de texto simples, não apenas Info. Para especificar texto que apareça somente em Info e não em texto simples, use `@ifnotplaintext`, assim:

```
@ifinfo
@ifnotplaintext
Isto estará no Info, mas não em texto simples.
@end ifnotplaintext
@end ifinfo
```

As regiões delimitadas por esses comandos são fontes Texinfo comuns, como em `@iftex`, e não fontes de formatador bruto, como em `@tex` (veja Seção 16.3 [Comandos do Formatador Bruto], Página 130).

16.3 Comandos do Formatador Bruto

Os Condicionais `@if...` recém descritos precisam ser usados somente com fonte normal do Texinfo. Por exemplo, a maioria dos recursos do TEX simples não funcionará dentro do `@iftex`. O propósito do `@if...` é o de fornecer processamento condicional para fonte do Texinfo, não fornecer acesso aos recursos de formatação subjacentes. Para isso, Texinfo fornece os assim chamados *comandos do formatador bruto*. Eles deveriam ser usados somente quando verdadeiramente exigidos (a maioria dos documentos não precisa deles).

O primeiro comando do formatador bruto é o `@tex`. Você consegue entrar completamente no TEX simples e usar '\' nos comandos do TEX, delineando uma região com os comandos `@tex` e `@end tex`. Todos os comandos e códigos de categoria do TEX simples são restaurados dentro de uma região do `@tex`. A única exceção é a de que o caractere `@` ainda introduz um comando, de forma que `@end tex` possa ser reconhecido. Os processadores do Texinfo não produzirão material em tal região, a menos que a saída TEX esteja sendo produzida.

Em casos complexos, você pode desejar definir novas macros do TEX dentro de `@tex`. Você precisa usar `\gdef` para fazer isso, não `\def`, porque as regiões de `@tex` são processadas em um grupo do TEX. Se você precisar fazer várias definições, você pode desejar configurar `\globaldefs=1` (o valor dele será restaurado para zero como de costume quando o grupo terminar em `@end tex`, de forma que não causará problemas com o resto do documento).

Como um exemplo, aqui está uma equação exibida escrita em TEX simples:

```
@tex
$$ \chi^2 = \sum_{i=1}^N
         \left (y_i - (a + b x_i)
         \over \sigma_i\right)^2 $$
@end tex
```

A saída desse exemplo aparecerá somente em um manual impresso. Se estiver lendo isso em um formato não gerado pelo TEX, você não verá a equação que aparece no manual impresso.

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - (a + bx_i)}{\sigma_i} \right)^2$$

Analogamente, você pode usar `@ifhtml ... @end ifhtml` para delimitar o fonte do Texinfo a ser incluído somente na saída HTML, e `@html ... @end html` para uma região do HTML bruto.

Da mesma forma, você pode usar `@ifxml ... @end ifxml` para delimitar o fonte do Texinfo a ser incluído somente na saída XML, e `@xml ... @end xml` para uma região de XML bruto. Regiões de texto bruto em outros formatos também estarão presentes na saída XML, mas com proteção de caracteres XML e dentro de elementos correspondentes. Por exemplo, o texto HTML bruto:

```
@html
<br />
@end html
```

será incluído na saída XML como:

```
<html>
&lt;br /&gt;
</html>
```

Novamente, da mesma forma, você pode usar `@ifdocbook ... @end ifdocbook` para delimitar o fonte do Texinfo a ser incluído somente na saída do Docbook, e `@docbook ... @end docbook` para uma região do Docbook bruto.

O comportamento de novas linhas em regiões brutas não é especificado.

Em todos os casos, no processamento bruto, `@` retém o mesmo significado que no restante do documento. Assim, os processadores do Texinfo precisam reconhecer e até mesmo executar, até certo ponto, o conteúdo das regiões brutas, independentemente do formato final da saída. Portanto, especificar mudanças que afetem globalmente o documento dentro de uma região bruta leva a um comportamento imprevisível e geralmente indesejável. Por exemplo, usar o comando `@kbdinputstyle` dentro de uma região bruta é indefinido.

O remédio é simples: não faça isso. Use os comandos do formatador bruto para o propósito pretendido deles, de fornecer material diretamente no formato subjacente. Quando você simplesmente quiser dar especificações diferentes do Texinfo para formatos diferentes de saída, use os Condicionais `@if...` e permaneça na sintaxe do Texinfo.

16.4 Condicionais Inline: `@inline`, `@inlineifelse`, `@inlineraw`

Texinfo fornece um conjunto de comandos condicionais com argumentos fornecidos entre chaves:

`@inlinefmt{`*formato*`, `*texto*`}`
: Processa o *texto* do Texinfo se a saída *formato* estiver sendo gerada.

`@inlinefmtifelse{`*formato*`, `*texto-então*`, `*texto-docontrário*`}`
: Processa o *texto-então* do Texinfo se a saída *formato* estiver sendo gerada; caso contrário, processa *texto-docontrário*.

`@inlineraw{`*formato*`, `*texto*`}`
: Similar, porém para *texto* bruto (veja Seção 16.3 [Comandos do Formatador Bruto], Página 130).

Os nomes de *formato* suportados são:

```
docbook  html  info  plaintext  tex  xml
```

Por exemplo,

```
@inlinefmt{html, @emph{texto somente HTML}}
```

é quase equivalente a

```
@ifhtml
@emph{texto somente HTML}
@end ifhtml
```

exceto que nenhum espaço em branco é adicionado, como acontece no último caso (ambiente).

Nesses comandos, o espaço em branco é ignorado depois da vírgula que separa os argumentos, como de costume, mas *não* é ignorado no final de *texto*.

Para inserir um sinal de arroba literal, uma chave esquerda ou uma chave direita em um dos argumentos, você precisa usar os comandos alfabéticos `@atchar{}` (veja Seção 12.1.1 [Inserindo um Símbolo Arroba], Página 95) e `@lbracechar{}` ou `@rbracechar{}` (veja Seção 12.1.2 [Inserindo Chaves], Página 95), ou a análise se tornará confusa.

Com `@inlinefmtifelse`, também é necessário usar `@comma{}` para evitar confundir uma '`,`' no texto com o delimitador. Com `@inlinefmt` e `@inlineraw`, `@comma{}` não é exigido (embora seja aceitável usá-lo), pois esses comandos sempre tem exatamente dois argumentos.

Para TEX, o *texto* processado não pode conter comandos delimitados por nova linha. O texto a ser ignorado (por exemplo, para não-TEX) pode, no entanto.

Dois outros Condicionais `@inline...` complementam os comandos `@ifset` e `@ifclear`; veja-se a próxima seção.

16.5 Sinalizadores: `@set`, `@clear`, condicionais e `@value`

Você pode direcionar os comandos de formatação do Texinfo para formatar ou para ignorar partes de um arquivo do Texinfo com os comandos `@set`, `@clear`, `@ifset` e `@ifclear`.

Aqui estão breves descrições desses comandos. Vejam-se as seções a seguir para mais detalhes:

`@set` *sinalizador* [*valor*]
: Configura a variável *sinalizador* para o *valor* opcional, se especificado.

`@clear` *sinalizador*
: Desdefine a variável *sinalizador*, se ou não ela foi definida anteriormente.

`@ifset` *sinalizador*
: Se *sinalizador* estiver configurado, o texto até o próximo comando `@end ifset` será formatado. Se *sinalizador* estiver limpo, o texto até o comando `@end ifset` seguinte será ignorado.

`@inlineifset{`*sinalizador*`,` *texto*`}`
: Versão delimitada por chaves de `@ifset`.

`@ifclear` *sinalizador*
: Se *sinalizador* estiver configurado, o texto até o próximo comando `@end ifclear` será ignorado. Se *sinalizador* estiver limpo, o texto até o comando `@end ifclear` seguinte será formatado.

`@inlineifclear{`*sinalizador*`,` *texto*`}`
: Versão delimitada por chaves de `@ifclear`.

16.5.1 `@set` e `@value`

Use o comando `@set` para especificar um valor para um sinalizador, que posteriormente será expandido pelo comando `@value`.

Um nome de *sinalizador* (também conhecido como *variável*) é um identificador que começa com um alfanumérico, um '`-`' ou um '`_`'. Os caracteres subsequentes, se existirem, não podem ser espaços em branco, '`@`', chaves, colchetes angulares ou quaisquer de '`~``^+|`'; outros caracteres, como '`%`', possivelmente funcionem. No entanto, é melhor usar somente letras e numerais em um nome de sinalizador, não '`-`' ou '`_`' ou outros—eles funcionarão em alguns contextos, mas não em todos, devido às limitações no TEX.

O valor é o restante da linha de entrada e pode conter qualquer coisa. No entanto, diferentemente da maioria dos outros comandos que tomam o restante da linha como um valor, `@set` não precisa aparecer no começo de uma linha.

Escreva o comando `@set` assim:

```
@set foo Esta é uma sequência de caracteres.
```

Isso configura o valor do sinalizador `foo` para "Esta é uma sequência de caracteres.".

Os formatadores do Texinfo então substituem um comando `@value{`*sinalizador*`}` pela string para a qual *sinalizador* estiver configurado. Assim, quando `foo` for configurado como mostrado acima, os formatadores do Texinfo convertem isto:

```
@value{foo}
```

para isto:

```
Esta é uma sequência de caracteres.
```

Você pode escrever um comando `@value` dentro de um parágrafo; mas precisa escrever um comando `@set` em uma linha própria.

Se você escrever o comando `@set` assim:

```
@set foo
```

sem especificar uma string, o valor de `foo` será a string vazia.

Se você limpar um sinalizador configurado anteriormente com `@clear` *`sinalizador`*, um comando `@value{sinalizador}` subsequente informará um erro.

Por exemplo, se você configurar `foo` conforme segue:

```
@set quanto muito, muito, muito
```

então os formatadores transformam

```
É um dia @value{quanto} chuvoso.
```

into

```
É um dia muito, muito, muito chuvoso.
```

Se você escrever

```
@clear quanto
```

então os formatadores transformam

```
É um dia @value{quanto} chuvoso.
```

em

```
É um dia {Nenhum valor para "quanto"} chuvoso.
```

`@value` não pode ser usado confiavelmente como o argumento para um comando de acento (veja Seção 12.4 [Inserindo Acentos], Página 100). Por exemplo, isto falha:

```
@set minhaletra a
@'@value{minhaletra}
```

16.5.2 `@ifset` e `@ifclear`

Quando um *sinalizador* é configurado, os comandos de formatação do Texinfo formatam o texto entre pares subsequentes de comandos `@ifset` *`sinalizador`* e `@end ifset`. Quando o *sinalizador* é limpo, os comandos de formatação do Texinfo *não* formatam o texto. `@ifclear` opera analogamente.

Escreva o texto formatado condicionalmente entre comandos `@ifset` *`sinalizador`* e `@end ifset`, assim:

```
@ifset sinalizador
texto-condicional
@end ifset
```

Por exemplo, você pode criar um documento que tenha duas variantes, como um manual para um modelo 'grande' e para um 'pequeno':

```
Você pode usar esta máquina para desenterrar arbustos sem machucá-los.

@set grande

@ifset grande
Ela também pode desenterrar árvores adultas.
@end ifset

Lembre-se de replantar imediatamente ...
```

No exemplo, os comandos de formatação formatarão o texto entre `@ifset grande` e `@end ifset` porque o sinalizador `grande` está configurado.

Quando *sinalizador* é limpo, os comandos de formatação do Texinfo *não* formatam o texto entre `@ifset` *`sinalizador`* e `@end ifset`; esse texto é ignorado e não aparece, seja na saída impressa, ou na do Info.

Por exemplo, se você limpar o sinalizador do exemplo precedente escrevendo um comando `@clear grande` depois do comando `@set grande` (mas antes do texto condicional), então os

comandos de formatação do Texinfo ignorarão o texto entre os comandos `@ifset grande` e `@end ifset`. Na saída formatada, esse texto não aparece; tanto na saída impressa quanto na do Info, você vê somente as linhas que dizem: "Você pode usar esta máquina para desenterrar arbustos sem machucá-los. Lembre-se de replantar imediatamente . . .".

Se um sinalizador for limpo com um comando `@clear` *`sinalizador`*, então os comandos de formatação formatam o texto entre pares subsequentes de comandos `@ifclear` e `@end ifclear`. Mas se o sinalizador for configurado com `@set` *`sinalizador`*, então os comandos de formatação *não* formatam o texto entre um comando `@ifclear` e um `@end ifclear`; em vez disso, eles ignoram esse texto. Um comando `@ifclear` se parece com isto:

```
@ifclear sinalizador
```

16.5.3 `@inlineifset` e `@inlineifclear`

`@inlineifset` e `@inlineifclear` fornecem alternativas delimitadas por chaves para as formas `@ifset` e `@ifclear`, similares aos outros comandos `@inline...` (veja Seção 16.4 [Condicionais Inline], Página 131). As mesmas ressalvas acerca de análise de argumentos dadas ali se aplicam aqui também.

`@inlineifset{`*`variável, texto`*`}`
: Processa o *texto* do Texinfo se o sinalizador *variável* estiver definido.

`@inlineifclear{`*`variável, texto`*`}`
: Processa o *texto* do Texinfo se o sinalizador *variável* não estiver definido.

Exceto para a sintaxe, o comportamento geral e propósitos deles são os mesmos de `@ifset` e `@ifclear`, descritos na seção anterior.

16.5.4 Exemplo de `@value`

Você pode usar o comando `@value` para minimizar o número de lugares que precisa mudar ao registrar uma atualização para um manual. Veja Seção C.2 [Textos GNU de Amostra], Página 233, para o texto completo de um exemplo de uso disso para trabalhar com distribuições Automake.

Este exemplo foi adaptado a partir do *The GNU Make Manual*.

1. Configura os sinalizadores:

```
@set EDITION 0.35 Beta
@set VERSION 3.63 Beta
@set UPDATED 14 de agosto de 1992
@set UPDATE-MONTH Agosto de 1992
```

2. Escreva texto para a seção `@copying` (veja Seção 3.3.1 [`@copying`], Página 17):

```
@copying
Esta é a Edição @value{EDITION},
atualizada mais recentemente em @value{UPDATED},
do @cite{The GNU Make Manual},
para @code{make}, versão @value{VERSION}.

Copyright ...

Permissão é concedida ...
@end copying
```

3. Escreva texto para a página de título, para pessoas leitoras do manual impresso:

```
@titlepage
@title GNU Make
@subtitle Um Programa para Direcionar Recompilação
@subtitle Edição @value{EDITION}, ...
@subtitle @value{UPDATE-MONTH}
@page
@insertcopying
...
@end titlepage
```

(Em uma capa impressa, uma data listando o mês e o ano parece menos complicada que uma data listando o dia, bem como o mês e o ano).

4. Escreva texto para o nó Top, para as pessoas leitoras do arquivo Info:

```
@ifnottex
@node Top
@top Make

Esta é a Edição @value{EDITION},
atualizada mais recentemente em @value{UPDATED},
do @cite{The GNU Make Manual},
para @code{make}, versão @value{VERSION}.
@end ifnottex
```

Depois de você formatar o manual, as construções `@value` foram expandidas, de forma que a saída contém texto como este:

```
Esta é a Edição 0.35 Beta, atualizada mais recentemente em 14 de agosto de 1992,
do `The GNU Make Manual', para `make', versão 3.63 Beta.
```

Quando atualizar o manual, você muda somente os valores dos sinalizadores; você não precisa editar as três seções.

16.6 Testes para Comandos do Texinfo: `@ifcommanddefined`, `@ifcommandnotdefined`

Ocasionalmente, você possivelmente queira providenciar para que teu manual teste se um dado comando do Texinfo está disponível e (presumivelmente) fazer algum tipo de formatação residual se não estiver. Existem Condicionais `@ifcommanddefined` e `@ifcommandnotdefined` para fazer isso.

Por exemplo:

```
@ifcommanddefined node
Bom, @samp{@@node} está definido.
@end ifcommanddefined
```

produzirá o esperado 'Bom, '`@node`' está definido.'.

Esse condicional também considerará verdadeiros quaisquer novos comandos definidos pelo documento via `@macro`, `@alias`, `@definfoenclose` e `@def(code)index` (veja Capítulo 17 [Definindo Novos Comandos do Texinfo], Página 137). Ressalva: a implementação do TEX informa comandos internos do TEX, além de todos os comandos do Texinfo, como sendo "definidos"; a implementação do `makeinfo` é confiável a esse respeito, no entanto.

Você pode verificar o arquivo `NEWS` na distribuição do fonte do Texinfo e vinculado a partir da página inicial do Texinfo (`http://www.gnu.org/software/texinfo`) para ver quando um comando específico foi adicionado.

Esses Condicionais de verificação de comando foram adicionados eles próprios no Texinfo 5.0, lançado em 2013—décadas depois do início do Texinfo. Para a finalidade de testar se eles próprios estão disponíveis, o sinalizador predefinido `txicommandconditionals` pode ser testado, assim:

```
@ifset txicommandconditionals
@ifcommandnotdefined foobarnode
(Bom, @samp{@@foobarnode} não está definido).
@end ifcommandnotdefined
@end ifset
```

Como os sinalizadores (veja-se a seção anterior) foram adicionados no início da existência do Texinfo, não existe problema em supor que eles estejam disponíveis.

Nós recomendamos evitar esses testes sempre que possível—o que geralmente é o caso. Para muitos pacotes de software, é razoável que todos os(as) desenvolvedores(as) tenham uma determinada versão do Texinfo (ou mais recente) instalada, e, portanto, não existe razão para se preocupar com versões mais antigas. (É simples para qualquer um(a) baixar e instalar o código-fonte do Texinfo; ele não tem quaisquer dependências problemáticas).

A questão das versões do Texinfo geralmente não surge para usuários(as) finais. Com pacotes distribuídos corretamente, os(as) usuários(as) não precisam processar o manual do Texinfo simplesmente para construir e instalar o pacote; eles(as) conseguem usar arquivos pré-formatados de saída do Info (ou outros). Isso é desejável em geral, para evitar dependências desnecessárias entre pacotes (veja Seção "Releases" em *GNU Coding Standards*).

16.7 Aninhamento de Condicional

Condicionais podem ser aninhados; no entanto, os detalhes são um pouco complicados. A dificuldade vem com Condicionais falhos, como `@ifhtml` quando HTML não está sendo produzido, onde o texto incluído é para ser ignorado. No entanto, ele não é para ser *completamente* ignorado, pois é útil ter um `@ifset` dentro de outro, por exemplo—essa é uma maneira de incluir texto somente se duas condições forem atendidas. Aqui está um exemplo:

```
@ifset algumavarariável
@ifset outravarariável
Ambas, algumavarariável e outravarariável estão configuradas.
@end ifset
@ifclear outravarariável
algumavarariável está configurada, outravarariável não está.
@end ifclear
@end ifset
```

Tecnicamente, Texinfo exige que, para um Condicional com falha, o texto ignorado precisa ser aninhado corretamente com relação àquele Condicional com falha. Infelizmente, nem sempre é possível verificar se *todos* os Condicionais estão aninhados corretamente, porque então os processadores poderiam ter que interpretar completamente o texto ignorado, o que anula o propósito do comando. Aqui está um exemplo ilustrando essas regras:

```
@ifset a
@ifset b
@ifclear ok  - ok, ignorado
@end junky   - ok, ignorado
@end ifset
@c ERRADO - ausente @end ifset.
```

Finalmente, como mencionado acima, todos os comandos condicionais precisam estar em linhas próprias, sem texto (nem mesmo espaços) antes ou depois. Caso contrário, os processadores não conseguem determinar confiavelmente quais comandos considerar para fins de aninhamento.

17 Definindo Novos Comandos do Texinfo

Texinfo fornece várias maneiras para definir novos comandos (em todos os casos, não é recomendado tentar redefinir comandos existentes):

- Uma *macro* do Texinfo permite que você defina um novo comando do Texinfo como qualquer sequência de texto e (ou) comandos existentes (incluindo outras macros). A macro pode ter qualquer número de *parâmetros*—texto que você fornece cada vez que usa a macro.

  Aliás, essas macros não tem nada a ver com o comando `@defmac`, que serve para documentar macros na área de assunto do manual (veja Seção 14.1 [Modelos de Comando de Definição], Página 114).
- '`@alias`' é uma maneira conveniente para definir um novo nome para um comando existente.
- '`@definfoenclose`' permite que você defina novos comandos com saída personalizada para todos os formatos de saída não TEX.

De modo geral (não apenas para definir novos comandos), é possível invocar qualquer processador externo de macro e fazer com que o Texinfo reconheça as assim chamadas diretivas `#line` para informes de erros.

Se você quiser fazer substituição simples de texto, `@set` e `@value` são a abordagem mais simples (veja Seção 16.5 [`@set @clear @value`], Página 132).

17.1 Definindo Macros

Você usa o comando `@macro` do Texinfo para definir uma macro, como isto:

```
@macro nomemacro{parâmetro1, parâmetro2, ...}
texto ... \parâmetro1\ ...
@end macro
```

Os *parâmetros parâmetro1*, *parâmetro2*, ... correspondem aos argumentos fornecidos quando a macro for posteriormente usada no documento (descrito na próxima seção).

Para uma macro funcionar consistentemente com TEX, *nomemacro* precisa consistir inteiramente de letras: sem dígitos, hifens, sublinhados ou outros caracteres especiais. Portanto, nós recomendamos usar somente letras. Entretanto, `makeinfo` aceitará qualquer coisa consistente de alfanuméricos e (exceto como o primeiro caractere) '`-`'. O caractere '`_`' é excluído, de forma que as macros possam ser chamadas dentro de `@math` sem um espaço seguinte (veja Seção 12.7 [Inserindo Fórmulas Matemáticas], Página 102).

Se uma macro não precisar de parâmetros, você pode defini-la ou com uma lista vazia ('`@macro foo {}`') ou sem chaves ('`@macro foo`').

A definição ou *corpo* da macro pode conter a maioria dos comandos do Texinfo, incluindo invocações de macro. No entanto, uma definição de macro que define outra macro não funciona no TEX devido a limitações no projeto do `@macro`.

No corpo da macro, instâncias de um nome de parâmetro cercado por barras invertidas, como em '`\parâmetro1\`' no exemplo acima, são substituídas pelo argumento correspondente originário da invocação da macro. Você pode usar nomes de parâmetros qualquer número de vezes no corpo, incluindo zero.

Para obter um '`\`' na expansão da macro, use '`\\`'. Qualquer outro uso de '`\`' no corpo produz um aviso.

Os caracteres de nova linha depois da linha `@macro` e antes da linha `@end macro` são ignorados, isto é, não são incluídos no corpo da macro. Todos os outros espaços em branco são tratados de acordo com as regras usuais do Texinfo.

Para permitir que uma macro seja usada recursivamente, isto é, em um argumento para uma chamada para si mesma, você precisa defini-la com '`@rmacro`', assim:

```
@rmacro rmac {arg}
a\arg\b
@end rmacro
...
@rmac{1@rmac{texto}2}
```

Isso produz a saída 'a1atextob2b'. Com '`@macro`' em vez de '`@rmacro`', uma mensagem de erro é fornecida.

Você pode indefinir uma macro *foo* com `@unmacro` *foo*. Não é um erro indefinir uma macro que já está indefinida. Por exemplo:

```
@unmacro foo
```

17.2 Invocando Macros

Depois que uma macro estiver definida (veja-se a seção anterior), você pode *invocar* (usá-la) em teu documento assim:

`@`*nomemacro* `{`*arg1*`, `*arg2*`, ...}`

e o resultado será mais ou menos como se você tivesse digitado o corpo de *nomemacro* naquele ponto. Por exemplo:

```
@macro foo {p, q}
Junto: \p\ & \q\.
@end macro
@foo{a, b}
```

produz:

Junto: a & b.

Assim, os argumentos e parâmetros são separados por vírgulas e delimitados por chaves; qualquer espaço em branco depois (mas não antes) de uma vírgula é ignorado. As chaves são exigidas na invocação mesmo quando a macro não receba argumentos, consistente com outros comandos do Texinfo. Por exemplo:

```
@macro argless {}
Sem argumentos aqui.
@end macro
@argless{}
```

produz:

Sem argumentos aqui.

Passar argumentos de macro contendo vírgulas exige cuidado, pois vírgulas também separam os argumentos. Para incluir um caractere de vírgula em um argumento, o método mais confiável é o de usar o comando `@comma{}`. Para `makeinfo`, você também pode prefixar um caractere de barra invertida, como em '`\,`', mas isso não funciona com TEX.

Nem sempre é necessário se preocupar com vírgulas. Para facilitar o uso de macros, `makeinfo` implementa duas regras para *citação automática* em algumas circunstâncias:

1. Se uma macro receber somente um argumento, todas as vírgulas na invocação dela serão aspadas por padrão. Por exemplo:

```
@macro TENTEME{texto}
@strong{TENTEME: \texto\}
@end macro

@TENTEME{Um recurso interessante, embora possa ser perigoso}.
```

produzirá a seguinte saída

TENTEME: Um recurso interessante, embora possa ser perigoso.

E, de fato, pode. Ou seja, `makeinfo` não controla o número de argumentos passados para macros de um argumento, de forma que seja cuidadoso(a) quando você invocá-las.

2. Se uma invocação de macro incluir outro comando (incluindo uma invocação recursiva dela mesma), quaisquer vírgulas na(s) invocação(ões) de comando(s) aninhado(s) serão aspadas por padrão. Por exemplo, em

```
@say{@strong{Yes, I do}, person one}
```

a vírgula depois de '`Yes`' é implicitamente aspada. Aqui está outro exemplo, com uma macro recursiva:

```
@rmacro cat{a,b}
\a\\b\
@end rmacro

@cat{@cat{foo, bar}, baz}
```

produzirá a string '`foobarbaz`'.

3. Caso contrário, uma vírgula deveria ser explicitamente aspada, como acima, para ser tratada como parte de um argumento.

A própria barra invertida pode ser aspada em argumentos de macro com outra barra invertida. Por exemplo:

```
@nomemac {\\bleh}
```

passará o argumento '`\bleh`' para *nomemac*.

`makeinfo` também reconhece sequências '`\{`' e '`\}`' para chaves, mas elas não são reconhecidas pela implementação em TEX. No entanto, raramente deveria existir necessidade delas, pois elas são necessárias somente quando um argumento de macro contiver chaves desbalanceadas.

Se uma macro for definida para receber exatamente um argumento, ela pode ser invocada sem quaisquer chaves, recebendo toda a linha depois do nome da macro como argumento. Por exemplo:

```
@macro bar {p}
Duas vezes: \p\ & \p\.
@end macro
@bar aah
```

produz:

Duas vezes: aah & aah.

Nesses argumentos, não existe escapagem de caracteres especiais, de forma que cada '`\`' representa ele mesmo.

Se uma macro for definida para receber mais que um argumento, mas for chamada com somente um (entre chaves), os argumentos restantes serão configurados para a string vazia e nenhum erro será dado. Por exemplo:

```
@macro addtwo {p, q}
Ambos: \p\\q\.
@end macro
@addtwo{a}
```

produz simplesmente:

Ambos: a.

17.3 Detalhes e Ressalvas Acerca de Macro

Por projeto, expansão de macro não ocorre nos seguintes contextos no `makeinfo`:

- linhas `@macro` e `@unmacro`;
- linhas `@if...`, incluindo `@ifset` e similares;
- `@set`, `@clear`, `@value`;
- linhas `@clickstyle`;
- linhas `@end`.

Infelizmente, o TEX pode fazer alguma expansão nessas situações, possivelmente gerando erros.

Além disso, algumas construções relacionadas a macro causam problemas com TEX; algumas das ressalvas estão listadas abaixo. Portanto, se você obtiver erros relacionados a macro ao produzir a versão impressa de um manual, você pode tentar expandir as macros com `makeinfo` invocando `texi2dvi` com a opção '`-E`' (veja Seção 19.2 [Formatar com `texi2dvi`], Página 150). Ou, mais confiavelmente, evite completamente as macros do Texinfo e use uma linguagem projetada para processamento de macros, como M4 (veja Seção 17.6 [Processadores Externos de Macro], Página 143).

- Como mencionado anteriormente, os nomes de macro precisam consistir inteiramente de letras.
- Não é aconselhável redefinir nenhum nome de comando primitivo do TEX, simples ou Texinfo, como uma macro. Infelizmente, esse é um conjunto de nomes grande e aberto, e os possíveis erros resultantes são imprevisíveis.
- Argumentos para macros recebendo mais que um argumento não podem cruzar linhas.
- Macros contendo um comando que precisa estar em uma linha própria, como um Condicional, não podem ser invocadas no meio de uma linha. Similarmente, macros contendo comandos orientados a linha ou texto, como ambientes `@example`, possivelmente se comportem imprevisivelmente no TEX.
- Comandos do Texinfo na expansão de uma macro no texto de uma entrada de índice podem acabar sendo tipografados como texto literal (incluindo um sinal "@"), em vez de serem interpretados com o significado pretendido deles.
- Espaços em branco são ignorados no início das linhas.
- Macros não podem ser usadas confiavelmente no argumento para comandos de acento (veja Seção 12.4 [Inserindo Acentos], Página 100).
- O escape de barra invertida para vírgulas em argumentos de macro não funciona; `@comma{}` precisa ser usado.
- Da mesma forma, se você quiser passar um argumento com o comando do Texinfo `@,` (para produzir uma cedilha, veja-se Seção 12.4 [Inserindo Acentos], Página 100), você tem que usar `@value` ou outra solução alternativa. Caso contrário, a vírgula pode ser tomada como separadora dos argumentos. Por exemplo,

```
@macro mactwo{argfirst, argsecond}
\argfirst\+\argsecond\.
@end macro
@set fc Fran@,cois
@mactwo{@value{fc},}
```

  produz:

  François+.

- Terminar um corpo de macro com '`@c`' pode fazer com que o texto após a invocação da macro seja ignorado como um comentário no `makeinfo`. Esse não é o caso ao processar com TEX (caso você realmente queira comentar o texto seguinte, use '`@comment`' em vez

disso). Isso era feito frequentemente para "comentar" uma nova linha indesejada no final de um corpo de macro, mas isso não mais é necessário, pois a nova linha final antes de '`@end macro`' não é incluída no corpo da macro de qualquer maneira.

- Em geral, você não pode substituir arbitrariamente uma chamada de macro (ou `@value`) por argumentos de comando do Texinfo, mesmo quando o texto é o mesmo. Texinfo não é M4 (nem mesmo TEX simples). Pode funcionar com alguns comandos, mas falha com outros. É melhor não fazer isso de jeito nenhum. Por exemplo, isto falha:

```
@macro offmacro
off
@end macro
@headings @offmacro
```

  Isso parece equivalente a `@headings off`, mas por motivos TEXnológicos, falha com uma mensagem de erro misteriosa (ou seja, '`Parágrafo terminou antes que @headings estivesse completo`').

- Macros não podem definir macros na maneira natural. Para fazer isso, você precisa usar Condicionais e TEX bruto. Por exemplo:

```
@ifnottex
@macro ctor {nome, arg}
@macro \nome\
algo envolvendo \arg\ de alguma forma
@end macro
@end macro
@end ifnottex
@tex
\gdef\ctor#1{\ctorx#1,}
\gdef\ctorx#1,#2,{\def#1{algo envolvendo #2 de alguma forma}}
@end tex
```

A implementação do `makeinfo` também tem as seguintes limitações (por projeto):

- `@verbatim` e macros não se misturam; por exemplo, você não pode iniciar um bloco verbatim dentro de uma macro e terminá-lo fora (veja Seção 8.5 [`@verbatim`], Página 69). Iniciar qualquer ambiente dentro de uma macro e terminá-lo fora pode ou não funcionar, nesse caso.
- Macros que definem macros completamente são ok, mas não é possível ter definições de macro aninhadas incompletamente. Ou seja, `@macro` e `@end macro` (da mesma forma para `@rmacro`) precisam ser pareados corretamente. Por exemplo, você não pode iniciar uma definição de macro dentro de uma macro e, em seguida, terminar essa definição aninhada fora da macro.

Na implementação do `makeinfo` antes do Texinfo 5.0, os fins de linhas originários da expansão de uma definição do `@macro` não terminavam um argumento delimitado por linha do comando @ (`@chapter`, `@center`, etc.). Esse não mais é o caso. Por exemplo:

```
@macro duaslinhas{}
aaa
bbb
@end macro
@center @duaslinhas{}
```

No `makeinfo` atual, isso é equivalente a:

```
@center aaa
bbb
```

com apenas '`aaa`' como argumento para `@center`. Na implementação anterior, ele teria sido analisado assim:

```
@center aaa bbb
```

17.4 '`@alias` *novo=existente*'

O comando '`@alias`' define um novo comando para ser exatamente como um existente. Isso é útil para definir nomes adicionais de marcação, preservando assim informações semânticas adicionais na entrada, mesmo que o resultado da saída possa ser o mesmo.

Escreva o comando '`@alias`' em uma linha própria, seguido pelo novo nome do comando, um sinal de igual e o nome do comando existente. O espaço em branco ao redor do sinal de igual é opcional e ignorado se presente. Assim:

```
@alias novo = existente
```

Por exemplo, se teu documento contiver citações para livros e algumas outras mídias (filmes, por exemplo), você pode querer definir uma macro `@moviecite{}` que faz a mesma coisa que um `@cite{}` comum, mas transmite as informações semânticas extras também. Você faria isso conforme segue:

```
@alias moviecite = cite
```

Macros nem sempre tem o mesmo efeito que apelidos, devido a caprichos da análise de argumentos. Além disso, apelidos são muito mais simples de definir que macros. Então o comando não é redundante.

Infelizmente, não é possível apelidar ambientes do Texinfo; por exemplo, `@alias lang=exemplo` é um erro.

Apelidos precisam não serem recursivos, direta ou indiretamente.

Não é aconselhável redefinir nenhum nome de comando do TEX primitivo, do TEX simples ou do Texinfo como um apelido. Infelizmente, esse é um conjunto muito grande de nomes, e os possíveis erros resultantes oriundos do TEX são imprevisíveis.

`makeinfo` aceitará os mesmos identificadores para apelidos que aceita para nomes de macro, isto é, alfanuméricos e (exceto como o primeiro caractere) '`-`'.

17.5 `@definfoenclose`: Destaque Personalizado

Um comando `@definfoenclose` pode ser usado para definir um comando de realce para todos os formatos de saída não TEX. Um comando definido usando `@definfoenclose` marca o texto colocando-o entre strings que precedem e seguem o texto. Você pode usar isso para obter um controle mais próximo da tua saída.

Presumivelmente, se você definir um comando com `@definfoenclose`, você criará um comando correspondente para TEX, ou em `texinfo.tex`, `texinfo.cnf`, ou dentro de um '`@iftex`' ou '`@tex`' no teu documento.

Escreva um comando `@definfoenclose` no início de uma linha seguido por três argumentos separados por vírgula. O primeiro argumento para `@definfoenclose` é o nome do comando @ (sem o `@`); o segundo argumento é a string delimitadora inicial; e o terceiro argumento é a string delimitadora final. Os dois últimos argumentos cercam o texto destacado na saída.

Uma string delimitadora pode conter espaços. Nem o delimitador inicial nem o final são exigidos. Se você não quiser um delimitador inicial, mas quiser um delimitador final, você precisa seguir o nome do comando com duas vírgulas em uma linha; do contrário, a string delimitadora final que você pretendia será naturalmente (mal)interpretada como a string delimitadora inicial.

Se você fizer um `@definfoenclose` no nome de um comando predefinido (como `@emph`, `@strong`, `@t` ou `@i`), a definição de invólucro substituirá a definição interna. Nós não recomendamos isso.

Um comando de invólucro definido dessa maneira recebe um argumento entre chaves, pois é destinado para novos comandos de marcação (veja Capítulo 7 [Marcando Texto], Página 56).

Por exemplo, você pode escrever:

```
@definfoenclose phoo,//,\\
```

perto do início de um arquivo do Texinfo para definir `@phoo` como um comando de formatação do Info que insere '//' antes e '\\' depois do argumento para `@phoo`. Você pode então escrever `@phoo{bar}` onde quiser que '//bar\\' seja realçado no Info.

Para formatação do TEX, você poderia escrever

```
@iftex
@global@let@phoo=@i
@end iftex
```

para definir `@phoo` como um comando que faz com que o TEX tipografe o argumento para `@phoo` em itálico.

Cada definição se aplica ao próprio formatador dela: uma para TEX, o outra para todo o resto. Os comandos do TEX bruto precisam estar em '`@iftex`'. O comando `@definfoenclose` não precisa estar em '`@ifinfo`', a menos que você queira usar definições diferentes para formatos de saída diferentes.

Aqui está outro exemplo: escreva

```
@definfoenclose headword, , :
```

perto do início do arquivo, para definir `@headword` como um comando de formatação do Info que insere nada antes e dois pontos depois do argumento para `@headword`.

Definições de '`@definfoenclose`' precisam não serem recursivas, direta ou indiretamente.

17.6 Processadores Externos de Macro: Diretivas de Linha

Macros do Texinfo (e os outros recursos dele de substituição de texto) funcionam bem em casos simples. Se teu documento precisar de processamento incomumente complexo, no entanto, a fragilidade e limitações delas podem ser um problema. Nesse caso, você pode querer usar um processador de macro completamente diferente, como M4 (veja *M4*) ou CPP (veja *The C Preprocessor*).

Com uma exceção, o Texinfo não precisa saber se a entrada dele é fonte "original" ou pre-processada a partir de algum outro arquivo fonte. Portanto, você pode organizar teu sistema de construção para invocar quaisquer programas que você goste para lidar com expansão de macro ou outras necessidades de pré-processamento. O Texinfo não oferece suporte interno para nenhum preprocessador em particular, já que nenhum programa parecia ser suficiente para as exigências de todos os documentos.

A única exceção são os números de linha em mensagens de erro. Nesse caso, o número de linha deveria se referir ao arquivo fonte original, seja ele qual for. Existe um mecanismo bem conhecido para isso: a assim chamada diretiva '`#line`'. O Texinfo suporta isso.

17.6.1 Diretiva '`#line`'

Uma linha de entrada como esta:

```
#line 100 "foo.ptexi"
```

indica que a próxima linha era a linha 100 do arquivo `foo.ptexi`, e, portanto, isso é ao que uma mensagem de erro deveria se referir. Tanto M4 (veja Seção "Preprocessor features" em *GNU M4*) quanto CPP (veja Seção "Line Control" em *The C Preprocessor* e Seção "Preprocessor Output" em *The C Preprocessor*) podem gerar tais linhas.

O programa `makeinfo` reconhece essas linhas por padrão, exceto dentro dos blocos `@verbatim` (veja Seção 8.5 [`@verbatim`], Página 69). O reconhecimento delas pode ser desativado completamente com `CPP_LINE_DIRECTIVES` (veja Seção 20.5.4 [Outras Variáveis de Personalização], Página 177), embora normalmente não haja razão para isso.

Para aqueles poucos programas (M4, CPP, Texinfo) que precisam documentar diretivas '`#line`' e, portanto, tem exemplos que, de outra forma, corresponderiam ao padrão, o comando `@hashchar{}` pode ser usado (veja Seção 12.1.5 [Inserindo um Símbolo Cerquilha], Página 96). A linha de exemplo acima se parece com isto no fonte para este manual:

```
@hashchar{}line 100 "foo.ptexi"
```

O comando `@hashchar` foi adicionado ao Texinfo em 2013. Se você não quiser depender dele, você também pode usar `@set` e `@value` para inserir o literal '`#`':

```
@set hash #
@value{hash}line 1 "example.c"
```

Ou, se adequado, um ambiente `@verbatim` pode ser usado em vez de `@example`. Como mencionado acima, o reconhecimento de `#line` é desabilitado dentro de blocos verbatim.

17.6.2 '`#line`' e TEX

Conforme mencionado, `makeinfo` reconhece as diretivas '`#line`' descritas na seção anterior. No entanto, `texinfo.tex` não reconhece e não consegue reconhecer. Portanto, tal linha será incorretamente tipografada literal se TEX a vir. A solução é a de usar as opções de expansão de macro do `makeinfo` antes de executar TEX. Existem três abordagens:

- Se você executar `texi2dvi` ou as variantes dele (veja Seção 19.2 [Formatar com `texi2dvi`], Página 150), poderá passar `-E` e `texi2dvi` executará `makeinfo` primeiro para expandir macros e eliminar '`#line`'.
- Se você executar `makeinfo` ou as variantes dele (veja Capítulo 20 [Tradutor Genérico `texi2any`], Página 162), poderá especificar `--no-ifinfo --iftex -E algumarquivo.out` e, em seguida, fornecer `algumarquivo.out` para `texi2dvi` em um comando separado.
- Ou você pode executar `makeinfo --dvi --Xopt -E`. (Ou `--pdf` em vez de `--dvi`). `makeinfo` então chamará `texi2dvi -E`.

Uma última ressalva relativa ao uso com TEX: como as diretivas `#line` não são reconhecidas, os números de linha emitidos pelo comando `@errormsg{}` (veja Seção 16.1 [Comandos Condicionais], Página 128), ou pelo próprio TEX, são os números de linha (incorretos) provenientes do arquivo derivado que o TEX está lendo, em vez dos números de linha especificados pelo preprocessador. Esse é outro exemplo do porque nós recomendamos executar `makeinfo` para os melhores diagnósticos (veja Seção 21.1.1 [Vantagens do `makeinfo`], Página 185).

17.6.3 Detalhes da Sintaxe '`#line`'

Detalhes de sintaxe para a diretiva '`#line`': o caractere '`#`' pode ser precedido ou seguido por um espaço em branco, a palavra '`line`' é opcional, e o nome do arquivo pode ser seguido por uma lista de inteiros separados por espaços em branco (esses são os assim chamados "sinalizadores" produzidos pelo CPP em alguns casos). Para aqueles(as) que gostam de saber os detalhes sangrentos, a expressão regular (Perl) real que é correspondida é esta:

```
/^\s*#\s*(line)? (\d+)(( "([^"]+)")(\s+\d+)*)?\s*$/
```

Até onde nós podemos dizer, os sinalizadores de inteiros finais somente ocorrem em conjunto com um nome de arquivo, de forma que isso é refletido na expressão regular.

Como exemplo, a seguir está uma diretiva '`#line`' sintaticamente válida, ou seja, a linha 1 de `/usr/include/stdio.h`:

```
# 1 "/usr/include/stdio.h" 2 3 4
```

Infelizmente, o nome de arquivo aspado (‘`"..."`’) tem que ser opcional, porque o M4 (especialmente) frequentemente pode gerar diretivas ‘`#line`’ dentro de um arquivo. Como o ‘`line`’ também é opcional, o resultado é que as linhas podem corresponder ao que você não esperaria, por exemplo,

```
# 1
```

As soluções possíveis estão descritas acima (veja Seção 17.6.1 [Diretiva ‘`#line`’], Página 143).

18 Arquivos de Inclusão

Quando um processador do Texinfo vê um comando `@include` em um arquivo do Texinfo, ele processa o conteúdo do arquivo nomeado pelo `@include` e o incorpora aos arquivos de saída sendo criados. Arquivos de inclusão permitem que você mantenha um documento grande como uma coleção de partes convenientemente pequenas.

18.1 Como Usar Arquivos de Inclusão

Para incluir outro arquivo dentro de um arquivo do Texinfo, escreva o comando `@include` no começo de uma linha e siga-o na mesma linha pelo nome de um arquivo a ser incluído. Por exemplo:

```
@include buffers.texi
```

comandos @ são expandidos em nomes de arquivo. O mais provável de ser útil é `@value` (veja Seção 16.5.1 [`@set @value`], Página 132), e mesmo assim somente em situações complicadas.

Um arquivo incluído deveria ser simplesmente um segmento de texto que você espera que seja incluído como está no arquivo geral do Texinfo ou no *exterior*; ele não deveria conter as partes padrão de início e fim de um arquivo do Texinfo. Em particular, você não deveria iniciar um arquivo incluído com uma linha dizendo '`\input texinfo`'; se fizer isso, esse texto será inserido no arquivo de saída literalmente. Da mesma forma, você não deveria terminar um arquivo incluído com um comando `@bye`; nada depois de `@bye` é formatado.

No passado distante, você era obrigado(a) a escrever uma linha `@setfilename` no início de um arquivo incluído, mas não mais. Agora, não importa se você escrever tal linha. Se uma linha `@setfilename` existir em um arquivo incluído, ela será ignorada.

18.2 `texinfo-multiple-files-update`

O modo Texinfo do GNU Emacs fornece o comando `texinfo-multiple-files-update`. Esse comando cria ou atualiza os ponteiros 'Próximo', 'Anterior' e 'Acima' dos arquivos incluídos, bem como aqueles no arquivo externo ou no geral do Texinfo, e cria ou atualiza um menu principal no arquivo externo. Dependendo se você o chama com argumentos opcionais, o comando atualiza somente os ponteiros na primeira linha `@node` dos arquivos incluídos ou todos eles:

`M-x texinfo-multiple-files-update`

Chamado sem quaisquer argumentos:

- Cria ou atualiza os ponteiros 'Próximo', 'Anterior' e 'Acima' da primeira linha `@node` em cada arquivo incluído em um arquivo externo ou em um geral do Texinfo.
- Cria ou atualiza os ponteiros de nó de nível 'Top' do arquivo externo ou do geral.
- Cria ou atualiza um menu principal no arquivo externo.

`C-u M-x texinfo-multiple-files-update`

Chamado com *`C-u`* como um argumento de prefixo:

- Cria ou atualiza ponteiros na primeira linha `@node` em cada arquivo incluído.
- Cria ou atualiza os ponteiros de nó de nível 'Top' do arquivo externo.
- Cria e insere um menu mestre no arquivo externo. O menu mestre é feito a partir de todos os menus em todos os arquivos incluídos.

`C-u 8 M-x texinfo-multiple-files-update`

Chamado com um argumento numérico de prefixo, como *`C-u 8`*:

- Cria ou atualiza *todos* os ponteiros 'Próximo', 'Anterior' e 'Acima' de todos os arquivos incluídos.

- Cria ou atualiza *todos* os menus de todos os arquivos incluídos.
- Cria ou atualiza os ponteiros de nó de nível 'Top' do arquivo externo ou do geral.
- E então cria um menu mestre no arquivo externo. Isso é similar a invocar `texinfo-master-menu` com um argumento quando você estiver trabalhando com apenas um arquivo.

Observe o uso do argumento de prefixo no uso interativo: com um argumento regular de prefixo, apenas `C-u`, o comando `texinfo-multiple-files-update` insere um menu mestre; com um argumento numérico de prefixo, como `C-u 8`, o comando atualiza *cada* ponteiro e cada menu em *todos* os arquivos e então insere um menu mestre.

18.3 Exigências dos Arquivos de Inclusão

Se você planeja usar o comando `texinfo-multiple-files-update`, o arquivo externo do Texinfo que lista os arquivos incluídos dentro dele deveria conter nada além das partes inicial e final de um arquivo do Texinfo, e uma série de comandos `@include` listando os arquivos incluídos. Ele não deveria nem incluir índices, que deveriam ser listados em um arquivo incluído próprio.

Além disso, cada um dos arquivos incluídos precisa conter exatamente um nó de nível mais alto (convencionalmente, `@chapter` ou equivalente), e esse nó precisa ser o primeiro nó no arquivo incluído. Além disso, cada um desses nós de nível mais alto em cada arquivo incluído precisa estar no mesmo nível hierárquico na estrutura do arquivo. Normalmente, cada um é um nó `@chapter`, um `@appendix` ou um `@unnumbered`. Assim, normalmente, cada arquivo incluído contém um, e somente um, nó de capítulo ou nível equivalente.

O arquivo externo deveria conter somente *um* nó, o nó 'Top'. Ele *não* deveria conter quaisquer nós além do nó único 'Top'. O comando `texinfo-multiple-files-update` não os processará.

18.4 Arquivo de Amostra com `@include`

Aqui está um exemplo de um arquivo externo do Texinfo com arquivos `@include` dentro dele antes de executar `texinfo-multiple-files-update`, que inseriria um menu principal ou um mestre:

```
\input texinfo @c -*-texinfo-*-
@settitle Exemplo de Inclusão

... Veja Apêndice C [Arquivos de Amostra do Texinfo], Página 232, para exemplos do rest
inicial ...

@ifnottex
@node Top
@top Exemplo de Inclusão
@end ifnottex

@include foo.texinfo
@include bar.texinfo
@include indice-conceitos.texinfo
@bye
```

Um arquivo incluído, como `foo.texinfo`, pode se parecer com isto:

```
@node Primeiro
@chapter Primeiro Capítulo

Conteúdo do primeiro capítulo ...
```

O conteúdo completo de `indice-conceitos.texinfo` pode ser tão simples quanto isto:

```
@node Índice de Conceitos
@unnumbered Índice de Conceitos

@printindex cp
```

O arquivo fonte externo do Texinfo para *The GNU Emacs Lisp Reference Manual* é chamado `elisp.texi`. Esse arquivo externo contém um menu mestre com 417 entradas e uma lista de 41 arquivos `@include`.

18.5 *arquivo* `@verbatiminclude`: Incluir um Arquivo Literal

Você pode incluir o conteúdo exato de um arquivo no documento com o comando `@verbatiminclude`:

```
@verbatiminclude nomearquivo
```

O conteúdo de *nomearquivo* é impresso em um ambiente literal (veja Seção 8.5 [`@verbatim`], Página 69). Geralmente, o arquivo é impresso exatamente como está, com todos os caracteres especiais e espaços em branco retidos. Nenhum recuo é adicionado; se você quiser recuo, cerque o `@verbatiminclude` dentro de `@example` (veja Seção 8.4 [`@example`], Página 68).

O nome do arquivo é tomado literalmente, com somente uma exceção: referências `@value{`*variável*`}` são expandidas. Isso torna possível incluir arquivos em outros diretórios dentro de uma distribuição, por exemplo:

```
@verbatiminclude @value{top_srcdir}/NEWS
```

(Você ainda tem de ter `top_srcdir` definido em primeiro lugar).

Para um método de impressão do conteúdo do arquivo em um tamanho menor de fonte, veja-se o final da seção acerca do `@verbatim`.

18.6 Evolução dos Arquivos de Inclusão

Quando o Info foi criado, era costume criar muitos arquivos Info pequenos acerca de um assunto. Cada arquivo do Info era formatado a partir do arquivo fonte do Texinfo dele próprio. Esse costume significava que o Emacs não precisava fazer um buffer grande para manter a íntegra de um grande arquivo do Info quando alguém queria informações; em vez disso, o Emacs alocava apenas memória suficiente para o pequeno arquivo do Info que continha as informações específicas buscadas. Dessa maneira, o Emacs podia evitar desperdício de memória.

As referências oriundas de um arquivo para outro foram feitas referindo-se ao nome do arquivo e também ao nome do nó. (Veja Seção 4.9.6 [Referenciando Outros Arquivos do Info], Página 38). Veja-se, também, (Seção 6.4.4 [`@xref` com quatro e cinco argumentos], Página 48).

Os arquivos de inclusão foram projetados principalmente como uma maneira para criar um manual impresso grande e unitário a partir de vários arquivos menores do Info. Em um manual impresso, todas as referências estavam dentro do mesmo documento, de forma que o TEX podia determinar automaticamente os números de página das referências. Os comandos de formatação do Info costumavam incluir arquivos somente para criar índices conjuntos; cada um dos arquivos individuais do Texinfo tinha que ser formatado para Info individualmente. (Cada um, portanto, exigia a linha `@setfilename` própria dele).

Entretanto, como arquivos grandes do Info agora são divididos automaticamente, não mais é necessário mantê-los pequenos.

Hoje em dia, vários arquivos do Texinfo são usados principalmente para documentos grandes, como o *The GNU Emacs Lisp Reference Manual*, e para projetos nos quais várias pessoas escrevem seções diferentes de um documento simultaneamente.

Além disso, os comandos de formatação do Info foram estendidos para funcionarem com o comando `@include`, tão somente para criar um arquivo grande do Info que seja dividido em arquivos menores, se necessário. Isso significa que você pode escrever menus e referências cruzadas sem nomear os diferentes arquivos do Texinfo.

19 Formatando e Imprimindo Cópia Impressa

Executar o comando `texi2dvi` ou `texi2pdf` é a maneira mais simples de criar uma saída imprimível. Esses comandos são instalados como parte do pacote do Texinfo.

Em mais detalhes, três principais comandos de shell são usados para imprimir saída formatada a partir de um manual do Texinfo: um converte o fonte do Texinfo em algo imprimível; um segundo ordena índices; e um terceiro efetivamente imprime o documento formatado. Quando usar os comandos de shell, você pode ou trabalhar diretamente no shell do sistema operacional ou trabalhar em um shell dentro do GNU Emacs (ou algum outro ambiente de computação).

Se estiver usando o GNU Emacs, você pode usar comandos fornecidos pelo modo Texinfo em vez de comandos de shell. Além dos três comandos para formatar um arquivo, ordenar os índices e imprimir o resultado, o modo Texinfo oferece atalhos de teclado para comandos para recentralizar o buffer de saída, mostrar a fila de impressão e deletar um trabalho da fila de impressão.

Detalhes estão nas seções seguintes.

19.1 Use TEX

O programa de tipografar chamado TEX é usado para formatar um documento do Texinfo para saída imprimível. TEX é um programa de tipografia muito poderoso e, quando usado corretamente, faz um trabalho excepcionalmente bom.

Veja Seção 19.16 [Obtendo TEX], Página 161, para informações acerca de como obter TEX. Ele não está incluso no pacote do Texinfo, sendo uma vasta suíte de software em si.

19.2 Formatar com `texi2dvi`

O programa `texi2dvi` cuida de todos os passos para produzir um arquivo DVI do TEX a partir de um documento do Texinfo. Similarmente, `texi2pdf` produz um arquivo PDF.

Para executar `texi2dvi` ou `texi2pdf` em um arquivo de entrada `foo.texi`, faça isto (onde '`prompt$ `' é o prompt do teu shell):

```
prompt$ texi2dvi foo.texi
prompt$ texi2pdf foo.texi
```

Conforme mostrado nesse exemplo, os nomes dos arquivos de entrada para `texi2dvi` e `texi2pdf` precisam incluir qualquer extensão, como '`.texi`'. (Sob MS-DOS e talvez em outras circunstâncias, você possivelmente necessite executar '`sh texi2dvi foo.texi`' em vez de depender do sistema operacional para invocar o shell no script '`texi2dvi`').

Para uma lista de todas as opções, execute '`texi2dvi --help`'. Algumas das opções são discutidas abaixo.

Com a opção `--pdf`, `texi2dvi` produz saída PDF em vez de DVI (veja Seção 19.15 [Saída PDF], Página 161), executando `pdftex` em vez de `tex`. Alternativamente, o comando `texi2pdf` é uma abreviação para executar '`texi2dvi --pdf`'. O comando `pdftexi2dvi` também é fornecido como uma conveniência para AUC-TEX (veja *AUC-TEX*, pois ele prefere meramente antepor '`pdf`' às ferramentas de produção de DVI para ter ferramentas de produção de PDF.

Com a opção `--dvipdf`, `texi2dvi` produz saída PDF executando TEX e então um programa DVI-para-PDF: se a variável de ambiente `DVIPDF` estiver configurada, esse valor será usado, senão o primeiro programa existente entre `dvipdfmx`, `dvipdfm`, `dvipdf`, `dvi2pdf`, `dvitopdf`. Esse método geralmente suporta melhor a tipografia de CJK que o `pdftex`.

Com a opção `--ps`, `texi2dvi` produz PostScript em vez de DVI, executando `tex` e então `dvips` (veja *Dvips*). (Ou o valor da variável de ambiente `DVIPS`, se configurada).

`texi2dvi` também pode ser usado para processar arquivos LaTeX. Normalmente, `texi2dvi` é capaz de adivinhar o idioma do arquivo de entrada pelo conteúdo e extensão do nome do arquivo dele; no entanto, se ele adivinhar errado, você pode especificar explicitamente o idioma de entrada usando a opção de linha de comando `--language=`*idioma*, onde *idioma* é ou '`latex`' ou '`texinfo`'.

Uma opção útil para `texi2dvi` é '`--command=`*comando*'. Isso insere *comando* em uma linha própria, depois de uma linha `@setfilename` em uma cópia temporária do arquivo de entrada, antes de executar o TeX. Com isso, você pode especificar diferentes formatos de impressão, tais como `@smallbook` (veja Seção 19.11 [@smallbook], Página 159), `@afourpaper` (veja Seção 19.12 [Papel A4], Página 159) ou `@pagesizes` (veja Seção 19.13 [@pagesizes], Página 160), sem realmente mudar o fonte do documento. (Você também pode fazer isso em todo o site com `texinfo.cnf`; veja Seção 19.9 [Preparando para TeX], Página 157).

A opção `-E` (equivalentemente, `-e` e `--expand`) faz a expansão de macro do Texinfo usando `makeinfo` em vez da implementação do TeX (veja Seção 17.3 [Detalhes de Macro], Página 140). Cada implementação tem limitações e vantagens próprias delas. Se essa opção for usada, nenhuma linha no arquivo fonte pode começar com a string `@c _texi2dvi` ou com a string `@c (_texi2dvi)`.

`texi2dvi` usa a opção `--build=`*modo* para especificar onde a compilação do TeX acontece e, como consequência, como os arquivos auxiliares são tratados. O modo de construção também pode ser configurado usando a variável de ambiente `TEXI2DVI_BUILD_MODE`. Os valores válidos para *modo* são:

'`local`' Compile no diretório atual, deixando todos os arquivos auxiliares por aí. Esse é o uso tradicional do TeX.

'`tidy`' Compile em um diretório local `*.t2d`, onde os arquivos auxiliares são deixados. Os arquivos de saída são copiados de volta para o arquivo original.

Usar o modo '`tidy`' traz várias vantagens:

- o diretório atual não fica abarrotado com uma abundância de arquivos temporários.
- a desordem pode ser ainda mais reduzida usando `--build-dir=diretório`: todos os diretórios `*.t2d` são armazenados lá.
- a desordem pode ser reduzida a zero usando, por exemplo, `--build-dir=/tmp/\$USER.t2d` ou `--build-dir=\$HOME/.t2d`.
- o arquivo de saída é atualizado depois de cada execução bem-sucedida do TeX, para fins de visualização simultânea da saída gerada. Em uma construção '`local`', o visualizador para durante a execução inteira do TeX.
- se a compilação falhar, o estado anterior do arquivo de saída será preservado.
- As compilações PDF e DVI são mantidas em subdiretórios separados, evitando qualquer possibilidade de incompatibilidade de arquivos auxiliares.

Por outro lado, como a compilação '`tidy`' ocorre em outro diretório, ocasionalmente o TeX não conseguirá encontrar alguns arquivos (por exemplo, ao usar `\graphicspath`): nesse caso, use `-I` para especificar os diretórios adicionais a considerar.

'`clean`' O mesmo que '`tidy`', mas remove o diretório auxiliar depois. Cada compilação, portanto, exige o ciclo completo.

`texi2dvi` usará `etex` (ou `pdfetex`) se estiver disponível, porque ele roda mais rápido em alguns casos e fornece informações adicionais de rastreamento ao depurar `texinfo.tex`. No entanto, essa versão estendida do TeX não é exigida, e a saída DVI é idêntica. (Hoje em dia,

`pdftex` e `pdfetex` são exatamente os mesmos, mas nós ainda executamos `pdfetex` para atender a instalações antigas do TEX).

`texi2dvi` tenta detectar arquivos auxiliares gerados pelo TEX, seja usando a opção `-recorder` ou procurando por '`\openout`' no arquivo de registro que uma execução do TEX produz. Você pode controlar como `texi2dvi` faz isso com a variável de ambiente `TEXI2DVI_USE_RECORDER`. Os valores válidos são:

'`yes`' use a opção `-recorder`, sem verificações.

'`no`' procure por '`\openout`' no arquivo de registro, sem verificações.

'`yesmaybe`'
verifique se a opção `-recorder` é suportada e, se sim, use-a; caso contrário, verifique se o rastreamento de '`\openout`' no arquivo de registro é suportado e, se sim, use-o; caso contrário, será um erro.

'`nomaybe`' o mesmo que '`yesmaybe`', exceto que o rastreamento '`\openout`' no arquivo de registro é verificado primeiro.

O padrão é '`nomaybe`'. Essa variável de ambiente é fornecida para fins de solução de problemas e possivelmente mude ou desapareça no futuro.

19.3 Formatar com `tex/texindex`

Você pode fazer a formatação básica de um arquivo do Texinfo com o comando de shell `tex` seguido pelo nome do arquivo do Texinfo. Por exemplo:

```
tex foo.texi
```

TEX produzirá um *arquivo DVI*, bem como vários arquivos auxiliares contendo informações para índices, referências cruzadas, etc. O arquivo DVI (para arquivo *DeVice Independent*) pode ser impresso em praticamente qualquer dispositivo, talvez depois de uma conversão adicional (veja-se a seção anterior).

O comando de formatação `tex` em si não ordena os índices; ele escreve um arquivo de saída de dados de índices desordenados. Para gerar um índice impresso depois de executar o comando `tex`, você primeiro precisa de um índice ordenado para trabalhar. O comando `texindex` ordena índices. (`texi2dvi`, descrito na seção anterior, executa `tex` e `texindex` conforme necessário).

`tex` gera arquivos de índices desordenados sob nomes seguindo uma convenção padrão: o nome do teu arquivo de entrada principal com qualquer extensão '`.texi`' ou similar substituída pelo nome de duas letras do índice. Por exemplo, os arquivos de saída de índices brutos para o arquivo de entrada `foo.texi` seriam, por padrão, `foo.cp`, `foo.vr`, `foo.fn`, `foo.tp`, `foo.pg` e `foo.ky`. Esses são exatamente os argumentos a fornecer para `texindex`.

Em vez de especificar-se explicitamente todos os nomes de arquivos de índices desordenados, é comum usar '`??`' como curingas de shell e fornecer o comando neste formato:

```
texindex foo.??
```

Esse comando executará `texindex` em todos os arquivos de índices desordenados, incluindo quaisquer índices de duas letras que você mesmo(a) definiu usando `@defindex` ou `@defcodeindex`. Você pode seguramente executar '`texindex foo.??`', mesmo se existirem arquivos com extensões de duas letras que não sejam arquivos de índice, como '`foo.el`'. O comando `texindex` informa, mas, inobstante, ignora esses arquivos.

Para cada arquivo especificado, `texindex` gera um arquivo de índice ordenado cujo nome é feito anexando '`s`' ao nome do arquivo de entrada; por exemplo, `foo.cps` é feito a partir de `foo.cp`. O comando `@printindex` procura um arquivo com esse nome (veja Seção 11.4 [Imprimindo Índices e Menus], Página 91). TEX não lê o arquivo de saída de índice bruto e `texindex` não o altera.

Depois de ter ordenado os índices, você precisa reexecutar `tex` sobre o arquivo do Texinfo. Isso regenera o arquivo de saída, dessa vez com entradas de índices atualizadas.

Por fim, você possivelmente precise executar `tex` mais uma vez para corrigir os números de página nas referências cruzadas.

Para resumir, este é um processo de cinco etapas. (Alternativamente, é um processo de uma etapa: execute `texi2dvi`; veja-se a seção anterior).

1. Execute `tex` sobre teu arquivo do Texinfo. Isso gera um arquivo DVI (com referências cruzadas indefinidas e sem índices) e os arquivos de índices brutos (com extensões de duas letras).
2. Execute `texindex` sobre os arquivos de índices brutos. Isso cria os correspondentes arquivos de índices ordenados (com extensões de três letras).
3. Execute `tex` novamente sobre teu arquivo do Texinfo. Isso regenera o arquivo DVI, dessa vez com índices e referências cruzadas definidas, mas com números de página para as referências cruzadas oriundos da execução anterior, geralmente incorretos.
4. Ordene os índices novamente, com `texindex`.
5. Execute `tex` uma última vez. Desta vez, os números de página corretos serão escritos para as referências cruzadas.

19.3.1 Formatando Documentos Parciais

Ocasionalmente, você possivelmente deseje imprimir um documento quando sabe que ele está incompleto, ou imprimir apenas um capítulo de um documento. Nesse caso, os arquivos auxiliares usuais que o TEX cria e os avisos que o TEX dá acerca de referências cruzadas indefinidas são apenas incômodos. Você consegue evitá-los com o comando `@novalidate`, que você precisa fornecer *antes* de quaisquer comandos de seccionamento ou referência cruzada.

Assim, o início do teu arquivo aparentaria aproximadamente assim:

```
\input texinfo
@novalidate
...
```

`@novalidate` também desativa a validação no `makeinfo`, assim como a opção `--no-validate` dele (veja Seção 20.4 [Validação de Ponteiro], Página 169).

Além disso, você não precisa executar `texindex` toda vez depois de executar `tex`. O comando de formatação `tex` simplesmente usa quaisquer arquivos de índices ordenados que existam provenientes de um uso anterior do `texindex`. Se eles estiverem desatualizados, isso geralmente é aceitável enquanto você estiver criando ou depurando um documento.

19.3.2 Detalhes do `texindex`

Na versão 6 do Texinfo, lançada em 2015, o programa `texindex` foi completamente reimplementado. A principal diferença funcional é a de que entradas de índice iniciando com uma chave esquerda ou chave direita (‘`{`’ respectivamente ‘`}`’) conseguem funcionar corretamente. Por exemplo, essas simples entradas de índice são processadas corretamente, incluindo o “index initial” mostrado no índice:

```
@cindex @{
@cindex @}
...
@printindex cp
```

Entretanto, para habilitar esse comportamento, é necessário (por enquanto) fornecer uma opção especial para o TEX no início de um documento fonte:

```
@tex
```

```
\global\usebracesinindexestrue
@end tex
```

Isso ocorre porque a implementação anterior do `texindex` abortava com uma mensagem de erro incorreta (‘`Nenhum número de página em \entry...`’) em tais entradas de índice quando manuseadas da maneira normal. Portanto, o TEX escrevia uma “ordenar string” incorreta usando o caractere ‘`|`’; isso não afetava o texto da entrada, mas a inicial do índice era o incorreto ‘`|`’, e a ordenação não era perfeita.

Por causa desse erro fatal, e porque relativamente poucos documentos tem entradas de índice iniciando com chaves, nós queremos fornecer algum tempo de transição para instalações terem o novo `texindex`. Em algum ponto no futuro, nós tornaremos `\usebracesinindexes` verdadeiro por padrão (o código do TEX acima continuará funcionando bem).

Embora não uma questão de funcionalidade, os(as) leitores(as) possivelmente estejam interessados(as) em saber que o novo `texindex` é um programa letrado (`http://en.wikipedia.org/wiki/Literate_programming`) usando Texinfo para documentação e `awk` (portável) para código. Um arquivo fonte, `texindex/ti.twjr` nesse caso, produz o programa executável, um documento imprimível e um documento online.

O sistema é chamado TexiWeb Jr. e foi criado por Arnold Robbins, que também escreveu o novo `texindex`. Não coincidentemente, ele também é o mantenedor de longa data do `gawk` (GNU Awk, veja *The GNU Awk User's Guide*). O arquivo `texindex/Makefile.am` mostra um exemplo de uso do sistema.

19.4 Imprimir com `lpr` a partir do Shell

A maneira de imprimir um arquivo DVI depende da instalação do teu sistema. Duas comuns são ‘`dvips foo.dvi -o`’ para criar um arquivo do PostScript primeiro e depois imprimi-lo; e ‘`lpr -d foo.dvi`’ para imprimir um arquivo DVI diretamente.

Por exemplo, os comandos seguintes (provavelmente) serão suficientes para ordenar os índices, formatar e imprimir este manual usando o script de shell `texi2dvi` (veja Seção 19.2 [Formatar com `texi2dvi`], Página 150).

```
texi2dvi texinfo.texi
dvips texinfo.dvi -o
lpr texinfo.ps
```

Dependendo da configuração do `lpr` na tua máquina, você poderá combinar as duas últimas etapas em `lpr -d texinfo.dvi`.

Você também pode gerar um arquivo PDF executando `texi2pdf` em vez de `texi2dvi`; um PDF frequentemente é imprimível diretamente. Ou você pode gerar um arquivo PCL usando `dvilj` em vez de `dvips`, se você tiver uma impressora que prefira esse formato.

`lpr` é um programa padrão em sistemas Unix, mas geralmente está ausente em MS-DOS/MS-Windows. Se for, basta criar um arquivo PostScript ou PDF ou PCL, o que for mais conveniente, e imprimi-lo da maneira usual para tua máquina (por exemplo, enviando para a porta apropriada, geralmente ‘`PRN`’).

19.5 Imprimindo a Partir de um Shell do Emacs

Você consegue fornecer comandos de formatação e de impressão a partir de um shell dentro do GNU Emacs, assim como qualquer outro comando de shell. Para criar um shell dentro do Emacs, digite *`M-x shell`* (veja Seção “Shell” em *O Manual do GNU Emacs*). Nesse shell, você consegue formatar e imprimir o documento. Veja Capítulo 19 [Formatar e Imprimir Cópia Impressa], Página 150, para detalhes.

Você consegue comutar para e do buffer do shell enquanto `tex` estiver executando e fazer outras edições. Se você estiver formatando um documento longo em uma máquina lenta, isso pode ser muito conveniente.

Por exemplo, você consegue usar `texi2dvi` a partir de um shell do Emacs. Aqui está uma maneira de usar `texi2pdf` para formatar e imprimir *Using and Porting GNU CC* a partir de um shell dentro do Emacs:

```
texi2pdf gcc.texi
lpr gcc.pdf
```

Veja-se a próxima seção para mais informações acerca de formatação e impressão no modo Texinfo.

19.6 Formatando e Imprimindo no Modo Texinfo

O modo Texinfo fornece vários comandos de tecla predefinidos para formatação e para impressão do TEX. Eles incluem comandos para ordenar índices, olhar a fila da impressora, terminar o trabalho de formatação e recentralizar a exibição do buffer no qual as operações ocorrem.

C-c C-t C-b
M-x texinfo-tex-buffer
: Execute `texi2dvi` no buffer atual.

C-c C-t C-r
M-x texinfo-tex-region
: Execute TEX na região atual.

C-c C-t C-i
M-x texinfo-texindex
: Ordene os índices de um arquivo do Texinfo formatado com `texinfo-tex-region`.

C-c C-t C-p
M-x texinfo-tex-print
: Imprima um arquivo DVI que foi criado com `texinfo-tex-region` ou `texinfo-tex-buffer`.

C-c C-t C-q
M-x tex-show-print-queue
: Mostre a fila de impressão.

C-c C-t C-d
M-x texinfo-delete-from-print-queue
: Delete um trabalho da fila de impressão; você será solicitado(a) a informar o número do trabalho mostrado por um comando *C-c C-t C-q* precedente (`texinfo-show-tex-print-queue`).

C-c C-t C-k
M-x tex-kill-job
: Termine o trabalho TEX executando atualmente iniciado por ou `texinfo-tex-region` ou `texinfo-tex-buffer`, ou qualquer outro processo executando no buffer do shell do Texinfo.

C-c C-t C-x
M-x texinfo-quit-job
: Saia de um trabalho de formatação do TEX que parou devido a um erro enviando um `x` para ele. Quando você faz isso, o TEX preserva um registro do que ele fez em um arquivo `.log`.

C-c C-t C-l
M-x tex-recenter-output-buffer
: Reexiba o buffer do shell no qual os comandos de impressão e de formatação do TEX forem executados para mostrar a saída mais recente deles.

Assim, a sequência usual de comandos para formatar um buffer é a seguinte (com comentários à direita):

`C-c C-t C-b`	Execute `texi2dvi` no buffer.
`C-c C-t C-p`	Imprima o arquivo DVI.
`C-c C-t C-q`	Exiba a fila da impressora.

Os comandos de formatação do modo Texinfo do TEX iniciam um subshell no Emacs chamado `*tex-shell*`. Os comandos `texinfo-tex-command`, `texinfo-texindex-command` e `tex-dvi-print-command` são todos executados nesse shell.

Você consegue observar os comandos operarem no buffer '`*tex-shell*`' e consegue comutar para e de, e usar o buffer '`*tex-shell*`' como faria com qualquer outro buffer de shell.

Os comandos de formatação e de impressão dependem dos valores de diversas variáveis. Os valores padrão são:

Variável	Valor padrão
`texinfo-texi2dvi-command`	`"texi2dvi"`
`texinfo-tex-command`	`"tex"`
`texinfo-texindex-command`	`"texindex"`
`texinfo-delete-from-print-queue-command`	`"lprm"`
`texinfo-tex-trailer`	`"@bye"`
`tex-start-of-header`	`"%**start"`
`tex-end-of-header`	`"%**end"`
`tex-dvi-print-command`	`"lpr -d"`
`tex-show-queue-command`	`"lpq"`

Você consegue mudar os valores dessas variáveis com o comando *M-x set-variable* (veja Seção "Examining and Setting Variables" em *O Manual do GNU Emacs*) ou com teu arquivo de inicialização `.emacs` (veja Seção "Init File" em *O Manual do GNU Emacs*).

Iniciando com a versão 20, o GNU Emacs oferece uma interface amigável, chamada *Customize*, para mudar valores de variáveis definidas pelo(a) usuário(a). Veja Seção "Easy Customization Interface" em *O Manual do GNU Emacs*, para mais detalhes acerca disso. As variáveis do Texinfo podem ser encontradas no grupo '`Development/Docs/Texinfo`', uma vez que você invoque o comando *M-x customize*.

19.7 Usando a Lista de Variáveis Locais

Outra maneira de aplicar o comando de formatação do TEX a um arquivo do Texinfo é a de colocar esse comando em uma *lista de variáveis locais* no final do arquivo do Texinfo. Você pode então especificar os comandos `tex` ou `texi2dvi` como um `compile-command` e fazer o Emacs executá-lo digitando *M-x compile*. Isso cria um shell especial chamado o buffer `*compilation*` no qual o Emacs executa o comando de compilar. Por exemplo, no final do arquivo `gdb.texi`, depois do `@bye`, você poderia colocar o seguinte:

```
Variáveis Locais:
compile-command: "texi2dvi gdb.texi"
End:
```

Essa técnica é mais frequentemente usada por programadores(as) que também compilam programas dessa maneira; veja-se Seção "Compilation" em *O Manual do GNU Emacs*.

19.8 Resumo das Exigências de Formatação do TEX

Cada arquivo do Texinfo que deva ser entrada para o TEX precisa iniciar com um comando `\input`:

```
\input texinfo
```

Isso instrui o TEX a carregar as macros que ele necessita para processar um arquivo do Texinfo.

Cada arquivo do Texinfo precisa terminar com uma linha que encerre o processamento do TEX e force a saída de páginas inacabadas:

```
@bye
```

Estritamente falando, essas duas linhas são tudo o que um arquivo do Texinfo precisa para ser processado com sucesso pelo TEX.

Geralmente, no entanto, o início inclui um comando `@settitle` para definir o título do manual impresso, uma página de título, uma página de direitos autorais, permissões e um tabela do conteúdo. Além de `@bye`, o fim de um arquivo geralmente inclui índices. (Sem mencionar que a maioria dos manuais também contém um corpo de texto).

Para mais informações, veja-se:

- Seção 3.2.4 [`@settitle`], Página 17.
- Seção 3.7.2 [`@setchapternewpage`], Página 25.
- Apêndice E [Cabeçalhos], Página 248.
- Seção 3.4 [Página de Título e Página de Direitos Autorais], Página 19.
- Seção 11.4 [Imprimindo Índices e Menus], Página 91.
- Seção 3.5 [Conteúdo], Página 22.

19.9 Preparando para TEX

TEX precisa saber onde encontrar o arquivo `texinfo.tex` que o comando '`\input texinfo`' na primeira linha lê. O arquivo `texinfo.tex` diz ao TEX como lidar com comandos @; ele está incluído em todas as distribuições GNU padrão. A versão mais recente lançada para uso geral está disponível a partir dos servidores e espelhos GNU usuais:

```
http://ftp.gnu.org/gnu/texinfo/texinfo.tex
http://ftpmirror.gnu.org/texinfo/texinfo.tex
```

A versão de desenvolvimento mais recente está disponível a partir do repositório de fonte do Texinfo:

```
http://svn.savannah.gnu.org/viewvc/trunk/doc/texinfo.tex?root=texinfo&view=log
```

`texinfo.tex` é essencialmente um arquivo independente, e a compatibilidade é uma preocupação extrema; então, se você precisa ou quer tentar uma versão mais nova que a que veio com teu sistema, quase sempre é suficiente baixá-la e colocá-la em qualquer lugar que o TEX a encontre (primeiro). Você pode substituir qualquer `texinfo.tex` existente por uma versão mais nova (claro, salvando a original em caso de desastre).

Além disso, você deveria instalar `epsf.tex`, se ele ainda não estiver instalado a partir de outra distribuição. Mais detalhes estão no final da descrição do comando `@image` (veja Seção 10.2 [Imagens], Página 84).

Para usar aspas diferentes daquelas usadas em inglês, você precisará ter as fontes European Computer Modern (por exemplo, `ecrm1000`) e (para saída em PDF) as fontes CM-Super (veja Seção 12.5 [Inserindo Aspas], Página 101).

Para usar o comando `@euro`, você precisará das fontes '`feym*`' (por exemplo, `feymr10`). Veja Seção 12.8.6 [`@euro`], Página 104.

Todos os arquivos acima (e muitos outros) deveriam ser instalados por padrão em uma instalação razoável do TEX.

Opcionalmente, você pode criar um arquivo `texinfo.cnf` para configuração do sítio. Esse arquivo é lido pelo TEX no início de um arquivo do Texinfo. Você pode colocar quaisquer comandos que gostar lá, de acordo com as convenções locais abrangentes a todo o sítio. Eles serão lidos pelo TEX ao processar qualquer documento do Texinfo. Por exemplo, se `texinfo.cnf` contiver a linha '`@afourpaper`' (veja Seção 19.12 [Papel A4], Página 159), então todos os documentos do Texinfo serão processados com esse tamanho de página em vigor. Se você não tiver nada para colocar em `texinfo.cnf`, não precisa criá-lo.

Se nenhum dos locais acima para esses arquivos de sistema for suficiente, você pode especificar os diretórios explicitamente. Para `texinfo.tex`, você pode fazer isso escrevendo o caminho completo para o arquivo depois do comando `\input`. Outra maneira, que funciona para ambos, `texinfo.tex` e `texinfo.cnf` (e qualquer outro arquivo que o TEX possa ler), é a de configurar a variável de ambiente `TEXINPUTS` no teu arquivo `.profile` ou `.cshrc`.

Se você usa `.profile` ou `.cshrc` depende se usa um interpretador de comandos compatível com o shell Bourne (`sh`, `bash`, `ksh`, ...) ou compatível com o shell C (`csh`, `tcsh`), respectivamente.

Em um arquivo `.profile`, você poderia usar a seguinte sequência de comandos do `sh`:

```
TEXINPUTS=.:/home/eu/minhabiblioteca:
export TEXINPUTS
```

Enquanto em um arquivo `.cshrc`, você poderia usar a seguinte sequência de comandos do `csh`:

```
setenv TEXINPUTS .:/home/eu/minhabiblioteca:
```

No MS-DOS/MS-Windows, você faria isto (observe o uso do caractere '`;`' como separador de diretório, em vez de '`:`'):

```
set TEXINPUTS=.;d:/home/eu/minhabiblioteca;c:
```

É costumeiro para usuários(as) de DOS/Windows coloquem tais comandos no arquivo `autoexec.bat` ou no registro do Windows.

Essas configurações fariam com que o TEX procurasse o arquivo `\input` primeiro no diretório atual, indicado por '`.`', depois no diretório `minhabiblioteca` de um(a) usuário(a) hipotético(a) '`eu`' e, finalmente, nos diretórios do sistema. (Um '`:`' anteposto, posposto ou duplo indica a busca nos diretórios do sistema naquele ponto).

19.10 "hboxes" Lotados

TEX ocasionalmente não consegue tipografar uma linha dentro das margens normais. Isso ocorre com mais frequência quando TEX encontra o que interpreta como uma palavra longa que ele não consegue hifenizar, como um endereço de rede de correio eletrônico ou um identificador muito longo. Quando isso acontece, TEX imprime uma mensagem de erro como esta:

```
Overfull @hbox (20.76302pt too wide)
```

(No TEX, as linhas estão em "caixas horizontais", daí o termo "hbox". '`@hbox`' é um primitivo do TEX não usado na linguagem do Texinfo).

TEX também fornece o número da linha no arquivo fonte do Texinfo e o texto da linha ofensiva, que é marcado em todos os lugares que TEX considerou hifenização. Veja Seção F.3 [Depuração com TEX], Página 253, para mais informações acerca de erros de tipografia.

Se o arquivo do Texinfo tiver um hbox lotado, você pode reescrever a frase, de forma que o hbox lotado não ocorra, ou você pode decidir deixá-lo. Uma pequena excursão na margem direita frequentemente não importa e possivelmente nem seja perceptível.

Se tiver muitas caixas lotadas e (ou) uma antipatia por reescrever, você pode forçar o TEX a aumentar bastante o espaçamento permitido entre palavras, evitando assim (se tiver sorte) muitas das quebras ruins de linha, como esta:

```
@tex
```

```
\global\emergencystretch = .9\hsize
@end tex
```

(Você deveria ajustar a fração conforme necessário). Esse valor enorme para `\emergencystretch` não pode ser o padrão, pois a saída tipografada geralmente seria de qualidade visivelmente inferior; o valor padrão dela é '`.15\hsize`'. `\hsize` é a dimensão do TEX que contém a largura atual da linha.

Para quaisquer caixas lotadas que você tenha, o TEX imprimirá um retângulo grande, feio e preto ao lado da linha que contém o hbox lotado, a menos que seja informado do contrário. Isso é para que você perceba o local do problema se estiver corrigindo um rascunho.

Para evitar que tal monstruosidade estrague tua impressão final, escreva o seguinte no início do arquivo do Texinfo, em uma linha própria, antes do comando `@titlepage`:

```
@finalout
```

19.11 `@smallbook`: Imprimindo Livros "Pequenos"

Por padrão, o TEX tipografa páginas para impressão em um formato de 8,5 por 11 polegadas. No entanto, você pode direcionar o TEX para tipografar um documento em um formato de 7 por 9,25 polegadas que seja adequado para livros encadernados inserindo o seguinte comando em uma linha própria no início do arquivo do Texinfo, antes da página de título:

```
@smallbook
```

(Como muitos livros tem cerca de 7 por 9,25 polegadas, esse comando poderia ter sido melhor chamado de comando `@regularbooksize`, mas veio a ser chamado de comando `@smallbook` por comparação ao formato de 8,5 por 11 polegadas).

Se você escrever o comando `@smallbook` entre as linhas de início de cabeçalho e fim de cabeçalho, o comando de formatação de região do modo Texinfo do TEX, `texinfo-tex-region`, formatará a região no tamanho de livro "pequeno" (veja Seção 3.2.2 [Início de Cabeçalho], Página 16).

Veja Seção 8.15 [`@small...`], Página 73, para informações acerca de comandos que facilitam produzir exemplos para um manual menor.

Veja Seção 19.2 [Formatar com `texi2dvi`], Página 150, e Seção 19.9 [Preparando para TEX], Página 157, para outras maneiras de formatar com `@smallbook` que não exigem mudar o arquivo fonte.

19.12 Imprimindo em Papel A4

Você pode dizer ao TEX para formatar um documento para impressão em papel A4 (ou A5) de tamanho europeu com o comando `@afourpaper` (ou `@afivepaper`). Escreva o comando em uma linha própria perto do início do arquivo Texinfo, antes da página de título. Por exemplo, isto é como você escreveria o cabeçalho para este manual:

```
\input texinfo    @c -*-texinfo-*-
@c %**start of header
@settitle Texinfo
@afourpaper
@c %**end of header
```

Veja Seção 19.2 [Formatar com `texi2dvi`], Página 150, e Seção 19.9 [Preparando para TEX], Página 157, para outras maneiras de formatar para diferentes tamanhos de papel que não exigem mudar o arquivo fonte.

Você pode ou não preferir a formatação que resulta do comando `@afourlatex`. Existe também `@afourwide` para papel A4 em formato largo.

19.13 `@pagesizes` [*largura*][, *altura*]: Tamanhos Personalizados de Página

Você pode especificar explicitamente a altura e (opcionalmente) a largura da área principal de texto na página com o comando `@pagesizes`. Escreva isso em uma linha própria perto do começo do arquivo do Texinfo, antes da página de título. A altura vem primeiro, depois a largura se desejado, separados por uma vírgula. Exemplos:

```
@pagesizes 200mm,150mm
```

e

```
@pagesizes 11.5in
```

Isso seria razoável para impressão em papel tamanho B5. Para enfatizar, esse comando especifica o tamanho da *área de texto*, não o tamanho do papel (que é 250 mm por 177 mm para B5, 14 in por 8,5 in para ofício).

Para fazer mudanças mais elaboradas, como mudar qualquer uma das margens da página, você precisa definir um novo comando no `texinfo.tex` ou `texinfo.cnf`.

Veja Seção 19.2 [Formatar com `texi2dvi`], Página 150, e Seção 19.9 [Preparando para TEX], Página 157, para outras maneiras de especificar `@pagesizes` que não exigem mudar o arquivo fonte.

19.14 Marcas de Corte e Ampliação

Você pode (tentar) direcionar o TEX para imprimir marcas de corte nos cantos das páginas com o comando `@cropmarks`. Escreva o comando `@cropmarks` em uma linha própria, perto do início do arquivo do Texinfo, antes da página de título, assim:

```
@cropmarks
```

Esse comando é principalmente para impressoras que tipografam várias páginas em uma folha de filme; mas, você pode tentar usá-lo para marcar os cantos de um livro configurado para 7 por 9,25 polegadas com o comando `@smallbook`. (As impressoras não produzirão marcas de corte para saída de tamanho regular que seja impressa em papel de tamanho regular). Como diferentes máquinas de impressão funcionam de maneiras diferentes, você deveria explorar o uso desse comando com um espírito de aventura. Você possivelmente tenha que redefinir o comando no `texinfo.tex`.

O comando `@cropmarks` é reconhecido e ignorado em formatos de saída não TEX.

Você pode tentar direcionar o TEX para tipografar páginas maiores ou menores que o normal com o comando `\mag` do TEX. Tudo que seja tipografado é dimensionado proporcionalmente maior ou menor. (`\mag` significa "ampliação"). Esse *não* é um comando @ do Texinfo, mas é um comando do TEX bruto que é prefixado com uma barra invertida. Você tem que escrever esse comando entre `@tex` e `@end tex` (veja Seção 16.3 [Comandos do Formatador Bruto], Página 130).

Siga o comando `\mag` com um '`=`' e então um número que seja 1.000 vezes a ampliação que você deseja. Por exemplo, para imprimir páginas em tamanho normal 1,2, escreva o seguinte próximo ao início do arquivo do Texinfo, antes da página de título:

```
@tex
\global\mag=1200
@end tex
```

Com algumas tecnologias de impressão, você pode imprimir cópias de tamanho normal que parecem melhores que o normal, entregando um mestre maior que o normal para tua gráfica. Eles(as) fazem a redução, aumentando assim efetivamente a resolução.

Dependendo do teu sistema, arquivos DVI preparados com um `\mag` não padrão podem não imprimir ou podem imprimir somente com certas ampliações. Esteja preparado(a) para experimentar.

19.15 Saída PDF

A maneira mais simples de gerar saída PDF a partir do fonte do Texinfo é a de executar o script de conveniência `texi2pdf` (ou `pdftexi2dvi`); isso executa o script `texi2dvi` com a opção `--pdf` (veja Seção 19.2 [Formatar com `texi2dvi`], Página 150). Se por algum motivo você quiser processar o documento manualmente, você pode executar o programa `pdftex` em vez do `tex` simples. Isto é, execute '`pdftex foo.texi`' em vez de '`tex foo.texi`'.

PDF significa 'Portable Document Format'. Foi inventado pela Adobe Systems alguns anos atrás para intercâmbio de documentos, baseados na linguagem PostScript deles. Links relacionados:

- GNU GV, um Leitor de PDF baseado em Ghostscript (`http://www.gnu.org/software/gv/`). (Ele também consegue pré visualizar documentos PostScript).
- `xpdf`, um leitor de PDF (`http://www.foolabs.com/xpdf/`) autônomo e disponível livremente para o sistema de janelas X.
- PDF na Wikipedia (`https://en.wikipedia.org/wiki/Portable_Document_Format`).

Atualmente, o Texinfo não fornece os comandos '`@ifpdf`' ou '`@pdf`' como para os outros formatos de saída, pois os documentos PDF contém muitos deslocamentos internos de baixo nível e referências cruzadas que seriam difíceis ou impossíveis de especificar no nível do fonte do Texinfo.

Arquivos PDF exigem software dedicado para serem exibidos, diferentemente dos formatos ASCII simples (Info, HTML) que o Texinfo suporta. Eles também tendem a ser muito maiores que os arquivos DVI produzidos pelo TEX por padrão. No entanto, um arquivo PDF define um documento tipografado real em um arquivo autocontido, notavelmente incluindo todas as fontes que são usadas, de forma que ele tem o lugar dele.

19.16 Obtendo TEX

TEX é um formatador de documentos que é usado pela FSF para a documentação dela. Ele é a maneira mais fácil de obter saída impressa (por exemplo, PDF e PostScript) para manuais do Texinfo. O TeX é redistribuível livremente, e você pode obtê-lo pela Internet ou em mídia física. Veja-se `http://tug.org/texlive`.

20 texi2any: O Tradutor Genérico para Texinfo

`texi2any` é o tradutor genérico para Texinfo que consegue produzir diferentes formatos de saída e é altamente personalizável. Ele suporta estes formatos:

Info (por padrão, ou com `--info`),
HTML (com `--html`),
texto simples (com `--plaintext`),
Docbook (com `--docbook`),
XML do Texinfo (com `--xml`).

`makeinfo` é um apelido para `texi2any`. Por padrão, tanto `texi2any` quanto `makeinfo` geram saída Info; de fato, não existem diferenças no comportamento baseadas no nome.

Além desses formatos padrão, as opções de linha de comando para `texi2any` podem mudar muitos aspectos da saída. Além disso, os arquivos de inicialização fornecem ainda mais controle sobre a saída final—quase tudo não especificado no arquivo de entrada do Texinfo. Os arquivos de inicialização são escritos em Perl, como o programa principal, e qualquer coisa que possa ser especificada na linha de comando também pode ser especificada dentro de um arquivo de inicialização.

O restante deste capítulo entra em detalhes.

20.1 texi2any: Uma Implementação de Referência do Texinfo

Acima, nós chamamos `texi2any` de "o" tradutor para Texinfo em vez de apenas "um" tradutor, embora (claro) seja técnica e juridicamente possível que outras implementações sejam escritas. O motivo é que implementações alternativas tem grande probabilidade de ter diferenças sutis, ou não tão sutis, no comportamento, e assim os documentos do Texinfo se tornariam dependentes do processador. Portanto, é importante ter uma implementação de referência que defina partes da linguagem não totalmente especificadas pelo manual (frequentemente intencionalmente). É igualmente importante ter opções de linha de comando consistentes e outros comportamentos para todos os processadores.

Por esse motivo, o processador Perl `texi2html` do Texinfo, antes independente, foi tornado compatível com a implementação C do `makeinfo`, para evitar continuar com duas implementações (veja Seção 1.6 [Histórico], Página 7). A implementação atual, `texi2any`, serve como a implementação de referência. Ela herdou o projeto de personalização e outros recursos de `texi2html` (para mais acerca de compatibilidade do `texi2html`, veja Seção 20.8 [`texi2html`], Página 183). No entanto, `texi2any` é uma reimplementação completa: ela constrói uma representação baseada em árvore do documento de entrada para todas as estruturas de retaguarda trabalharem.

Testes extensivos da linguagem foram desenvolvidos ao mesmo tempo que `texi2any`; nós apelamos a qualquer um(a) pensando em escrever um programa para analisar a entrada do Texinfo para, pelo menos, fazer uso desses testes.

O script envolucrador `texi2html` (veja Seção 20.8 [`texi2html`], Página 183) fornece um exemplo muito simples de chamada do `texi2any` a partir de um script de shell; ele está em `util/texi2html` nos fontes do Texinfo. Mais consequentemente, `texi-elements-by-size` é um script Perl de exemplo usando a interface do módulo `Texinfo::Parser`; ele também está no diretório `util` do fonte. (A funcionalidade dele também pode ser útil para autores(as); veja [texi-elements-by-size], Página 230).

Com o lançamento do `texi2any` como a implementação de referência, o desenvolvimento tanto da implementação C do `makeinfo` quanto do `texi2html` foi interrompido. Daqui para frente, nós pedimos para os(as) autores(as) de documentos do Texinfo que usem somente `texi2any`.

20.2 Invocando `texi2any`/`makeinfo` a partir de um Shell

Para processar um arquivo do Texinfo, invoque `texi2any` ou `makeinfo` (os dois nomes são sinônimos para o mesmo programa; nós usaremos os nomes intercambiavelmente) seguido pelo nome do arquivo do Texinfo. Selecione também o formato que você quer gerar com a opção apropriada de linha de comando (padrão é Info). Assim, para criar o arquivo Info para Bison, digite o seguinte para o shell:

```
texi2any --info bison.texinfo
```

Você pode especificar mais que um nome de arquivo de entrada; cada um é processado por vez. Se um nome de arquivo de entrada for '`-`', a entrada padrão será lida.

O programa `texi2any` aceita muitas opções. Talvez as mais básicas sejam aquelas que mudam o formato da saída gerada. Por padrão, `texi2any` produz Info.

Cada opção de linha de comando é ou um nome longo precedido por '`--`' ou uma letra precedida por '`-`'. Você pode usar abreviações para os nomes longos de opções, desde que elas sejam únicas.

Por exemplo, você poderia usar o seguinte comando de shell para criar um arquivo Info para `bison.texinfo` no qual as linhas sejam preenchidas com somente 68 colunas:

```
texi2any --fill-column=68 bison.texinfo
```

Você pode escrever duas ou mais opções em sequência, assim:

```
texi2any --no-split --fill-column=70 ...
```

(Isso manteria o arquivo Info unido como um arquivo possivelmente muito longo e também configuraria a coluna de preenchimento para 70).

As opções são (aproximadamente em ordem alfabética):

`--commands-in-node-names`
: Essa opção agora não faz nada, mas permanece para compatibilidade. (Ela costumava garantir que os comandos @ em nomes de nós fossem expandidos por todo o documento, especialmente `@value`. Isso agora é feito por padrão).

`--conf-dir=`*caminho*
: Pospõe *caminho* a lista de pesquisa de diretório para encontrar arquivos de personalização que possam ser carregados com `--init-file` (veja-se abaixo). O valor *caminho* pode ser um diretório ou uma lista de vários diretórios separados pelo caractere separador de caminho usual ('`:`' em sistemas do tipo Unix, '`;`' no Windows).

`--css-include=`*arquivo*
: Ao produzir HTML, inclui literalmente o conteúdo de *arquivo*, que deveria conter especificações do W3C de folhas de estilo em cascata, no bloco '`<style>`' da saída gerada em HTML. Se *arquivo* for '`-`', lê a entrada padrão. Veja Seção 22.3 [CSS de HTML], Página 195.

`--css-ref=`*url*
: Ao produzir HTML, adiciona uma etiqueta '`<link>`' à saída que referencia uma folha de estilo em cascata em *url*. Isso permite usar folhas de estilo autônomas.

`-D` *variável*
`-D '`*variável valor*`'`
: Faz com que a variável *variável* do Texinfo seja definida. Isso é equivalente a `@set` *variável* no arquivo do Texinfo (veja Seção 16.5 [`@set @clear @value`], Página 132).

  O argumento para a opção é sempre uma palavra para o shell; se contiver espaços em branco internos, a primeira palavra é tomada como o nome da variável e o restante como o valor. Por exemplo, `-D 'minhavariável algumvalor'` é equivalente a `@set minhavariável algumvalor`.

`--disable-encoding`
`--enable-encoding`
: Por padrão, ou com `--enable-encoding`, produz caracteres acentuados e especiais na saída Info e de texto simples baseada em '`@documentencoding`'. Com `--disable-encoding`, transliterações ASCII de 7 bits são produzidas. Veja Seção 15.2 [`@documentencoding`], Página 126, e Seção 12.4 [Inserindo Acentos], Página 100.

`--docbook`
: Gera saída Docbook (em vez de Info).

`--document-language=`*idioma*
: Usa *idioma* para traduzir palavras-chave do Texinfo que acabam no documento de saída. O padrão é a localidade especificada pelo comando `@documentlanguage` se existir um, caso contrário, inglês (veja Seção 15.1 [`@documentlanguage`], Página 125).

`--dvi` Gera um arquivo DVI do TeX usando `texi2dvi`, em vez de Info (veja Seção 20.3 [Saída Impressa do `texi2any`], Página 169).

`--dvipdf` Gera um arquivo PDF usando `texi2dvi --dvipdf`, em vez de Info (veja Seção 20.3 [Saída Impressa do `texi2any`], Página 169).

`--error-limit=`*limite*
`-e` *limite* Informa *limite* erros antes de abortar (supondo que continuar seria inútil); padrão 100.

`--fill-column=`*largura*
`-f` *largura*
: Especifica o número máximo de colunas em uma linha; essa é a borda direita de uma linha. Parágrafos que estejam preenchidos estarão preenchidos com essa largura. (Preenchimento é o processo de quebrar e conectar linhas, de forma que as linhas tenham o mesmo comprimento ou sejam menores que o número especificado como coluna de preenchimento. As linhas são quebradas entre palavras). O valor padrão é 72.

`--footnote-style=`*estilo*
`-s` *estilo* Configura o estilo da nota de rodapé como *estilo*: ou '`end`' para o estilo do nó final (o padrão) ou '`separate`' para o estilo do nó separado. O valor configurado por essa opção substitui o valor configurado em um arquivo do Texinfo por um comando `@footnotestyle` (veja Seção 10.3.2 [Estilos de Notas de Rodapé], Página 87).

Quando o estilo da nota de rodapé for '`separate`', `makeinfo` cria um novo nó contendo as notas de rodapé encontradas no nó atual. Quando o estilo da nota de rodapé for '`end`', `makeinfo` coloca as referências da nota de rodapé no final do nó atual.

Em HTML, quando o estilo de nota de rodapé for '`end`', ou se a saída não for dividida, as notas de rodapé serão colocadas no final da saída. Se configurada como '`separate`', e a saída for dividida, elas serão colocadas em um arquivo separado.

`--force`
`-F` Normalmente, se o arquivo de entrada tiver erros, os arquivos de saída não serão criados. Com essa opção, eles serão preservados.

`--help`
`-h` Imprime uma mensagem com as opções disponíveis e o uso básico e sai com sucesso.

`--html` Gera saída HTML (em vez de Info). Por padrão, a saída HTML é dividida em um arquivo de saída por nó do fonte do Texinfo, e a saída dividida é escrita em um

subdiretório baseado no nome do arquivo do Info de nível superior. Veja Capítulo 22 [Gerando HTML], Página 194.

`-I` *caminho*
: Pospõe *caminho* na lista de pesquisa de diretório para encontrar arquivos que estejam incluídos usando o comando `@include`. Por padrão, `texi2any` pesquisa somente o diretório atual. Se *caminho* não for fornecido, o diretório atual será posposto. O valor de *caminho* pode ser um diretório ou uma lista de vários diretórios separados pelo caractere usual separador de caminho (‘`:`’ em sistemas do tipo Unix, ‘`;`’ no Windows).

`--ifdocbook`
`--ifhtml`
`--ifinfo`
`--ifplaintext`
`--iftex`
`--ifxml`
: Para o formato fornecido, processa os comandos ‘`@if`*formato*’ e ‘`@`*formato*’ e não processa ‘`@ifnot`*formato*’, independentemente do formato sendo gerado. Por exemplo, se `--iftex` for fornecido, então os blocos ‘`@iftex`’ e ‘`@tex`’ serão lidos, e os blocos ‘`@ifnottex`’ serão ignorados.

`--info`
: Gera saída Info. Por padrão, se o arquivo de saída contiver mais que cerca de 300.000 bytes, ele será dividido em subarquivos menores, de cerca desse tamanho. O nome do arquivo de saída, e quaisquer subarquivos, é determinado por `@setfilename` (veja Seção 3.2.3 [`@setfilename`], Página 16). Veja Seção 21.1.5 [Arquivos de Etiqueta e de Divisão], Página 187.

`--init-file=`*arquivo*
: Carrega *arquivo* como código para modificar o comportamento e a saída do manual gerado. É costumeiro usar as extensões `.pm` ou `.init` para esses arquivos de personalização, mas isso não é obrigatório; o nome *arquivo* pode ser qualquer coisa. A opção `--conf-dir` (veja-se acima) pode ser usada para adicionar à lista de diretórios nos quais esses arquivos de personalização serão pesquisados.

`--internal-links=`*arquivo*
: No modo HTML, produz um arquivo separado por tabulações contendo três colunas: o link interno para um item indexado ou item no sumário; o nome do índice (ou sumário) no qual ele ocorre; e o termo que foi indexado ou inserido. Os itens estão na ordem natural de ordenação para o elemento fornecido. Esse despejo pode ser útil para pós-processadores.

`--macro-expand=`*arquivo*
`-E` *arquivo*
: Gera o fonte do Texinfo, com todas as macros do Texinfo expandidas, para *arquivo*. Normalmente, o resultado da expansão da macro é usado internamente por `makeinfo` e então descartado.

`--no-headers`
: Não inclui menus ou linhas separadoras de nós na saída gerada.

  Ao gerar Info, isso é o mesmo que usar `--plaintext`, resultando em um arquivo de texto simples. Além disso, `@setfilename` é ignorado, e a saída é para a saída padrão, a menos que substituída por `-o`. (Esse comportamento é para retro compatibilidade).

  Ao gerar HTML, e a saída for dividida, também produz links de navegação somente no começo de cada arquivo. Se a saída não for dividida, não inclui links de navegação no topo de cada nó. Veja Capítulo 22 [Gerando HTML], Página 194.

`--no-ifdocbook`
`--no-ifhtml`
`--no-ifinfo`
`--no-ifplaintext`
`--no-iftex`
`--no-ifxml`

Para o formato fornecido, não processa os comandos '`@if`*formato*' e '`@`*formato*' e processa '`@ifnot`*formato*', independentemente do formato sendo gerado. Por exemplo, se `--no-ifhtml` for fornecido, então os blocos '`@ifhtml`' e '`@html`' não serão lidos, e os blocos '`@ifnothtml`' serão.

`--no-node-files`
`--node-files`

Ao gerar HTML, cria arquivos de redirecionamento para âncoras e quaisquer nós ainda não gerados com o nome de arquivo correspondente ao nome do nó (veja Seção 22.4.2 [Expansão Xref de Nome de Nó do HTML], Página 198). Isso torna possível que referências entre manuais em nível de seção e de capítulo sejam bem-sucedidas (veja Seção 22.4.6 [Configuração do Xref do HTML], Página 201).

Se a saída for dividida, isso é habilitado por padrão. Se a saída não for dividida, `--node-files` habilita a criação dos arquivos de redirecionamento, além do arquivo de saída principal monolítico. `--no-node-files` suprime a criação de arquivos de redirecionamento em qualquer caso. Essa opção não tem efeito com nenhum formato de saída diferente de HTML. Veja Capítulo 22 [Gerando HTML], Página 194.

`--no-number-footnotes`

Suprime numeração automática de notas de rodapé. Por padrão, as notas de rodapé são numeradas sequencialmente dentro de um nó, ou seja, o número atual da nota de rodapé é reconfigurado para 1 no início de cada nó.

`--no-number-sections`
`--number-sections`

Com `--number_sections` (o padrão), gera números de capítulos, de seções e de anexos como em manuais impressos. Isso funciona somente com manuais estruturados hierarquicamente. Você deveria especificar `--no-number-sections` se teu manual não for normalmente estruturado.

`--no-pointer-validate`
`--no-validate`

Suprime a fase de validação de ponteiro de `makeinfo`—uma coisa perigosa de se fazer. Isso também pode ser feito com o comando `@novalidate` (veja Seção 19.1 [Use TEX], Página 150). Normalmente, verificações de consistência são feitas para garantir que referências cruzadas possam ser resolvidas, etc. Veja Seção 20.4 [Validação de Ponteiro], Página 169.

`--no-warn`

Suprime mensagens de aviso (mas não mensagens de erro).

`--output=`*arquivo*
`-o` *arquivo*

Especifica que a saída deveria ser direcionada para *arquivo*. Isso substitui qualquer nome de arquivo especificado em um comando `@setfilename` encontrado no fonte do Texinfo. Se nem `@setfilename` nem essa opção forem especificadas, o nome do arquivo de entrada será usado para determinar o nome de saída. Veja Seção 3.2.3 [`@setfilename`], Página 16.

Se *arquivo* for '`-`', a saída vai para a saída padrão e '`--no-split`' será implícito.

Se *arquivo* for um diretório ou terminar com '`/`', as regras usuais serão usadas para determinar o nome do arquivo de saída (ou seja, usar `@setfilename` ou o nome do arquivo de entrada), mas os arquivos serão escritos no diretório *arquivo*. Por exemplo, '`makeinfo -o bar/ foo.texi`', com ou sem `--no-split`, escreverá `bar/foo.info`, e possivelmente outros arquivos, sob `bar/`.

Ao gerar HTML e a saída for dividida, *arquivo* é usado como o nome para o diretório no qual todos os arquivos são escritos. Por exemplo, '`makeinfo -o bar --html foo.texi`' escreverá `bar/index.html`, entre outros arquivos.

`--output-indent=`*valor*
: Essa opção agora não faz nada, mas permanece para compatibilidade. (Ela costumava alterar o recuo na saída XML/Docbook).

`-P` *caminho*
: Antepõe *caminho* na lista de pesquisa de diretório para `@include`. Se *caminho* não for fornecido, o diretório atual será anteposto. Veja-se '`-I`' acima.

`--paragraph-indent=`*recuo*
`-p` *recuo*
: Configura o estilo de recuo do parágrafo para *recuo*. O valor configurado por essa opção substitui o valor configurado em um arquivo do Texinfo por um comando `@paragraphindent` (veja Seção 3.7.4 [`@paragraphindent`], Página 27). O valor de *recuo* é interpretado conforme segue:

  '`asis`'
  : Preserva qualquer recuo existente (ou falta dele) no início dos parágrafos.

  '`0`' ou '`none`'
  : Deleta qualquer recuo existente.

  número
  : Recua cada parágrafo com *número* espaços.

  O padrão é o de recuar dois espaços, exceto para parágrafos seguinte a um título de seção, os quais não são recuados.

`--pdf`
: Gera um arquivo PDF usando `texi2dvi --pdf`, em vez de Info (veja Seção 20.3 [Saída Impressa do `texi2any`], Página 169).

`--plaintext`
: Produz um arquivo de texto simples (em vez de Info): não inclui menus ou linhas separadoras de nós na saída. Isso resulta em um arquivo de texto simples direto que você pode (por exemplo) enviar em mensagem eletrônica sem complicações, ou incluir em uma distribuição (por exemplo, um arquivo `INSTALL`).

  Com essa opção, `@setfilename` é ignorado e a saída vai para a saída padrão por padrão; isso pode ser substituído por `-o`.

`--ps`
: Gera um arquivo PostScript usando `texi2dvi --ps`, em vez de Info (veja Seção 20.3 [Saída Impressa do `texi2any`], Página 169).

`--set-customization-variable` *variável=valor*
`-c` *variável=valor*
: Configura a variável de personalização *variável* para *valor*. O `=` é opcional, mas tanto *variável* quanto *valor* precisam ser aspados para o shell, conforme necessário, de forma que o resultado seja uma palavra. Muitos aspectos do comportamento e da saída de `texi2any` podem ser controlados por variáveis de personalização, além do que pode ser configurado no documento por comandos @ e com outras chaves de linha de comando. Veja Seção 20.5 [Variáveis de Personalização], Página 170.

`--split=`*como*
`--no-split`
: Ao gerar Info, por padrão, arquivos de saída grandes são divididos em subarquivos menores, de aproximadamente 300k bytes. Ao gerar HTML, por padrão, cada arquivo de saída contém um nó (veja Capítulo 22 [Gerando HTML], Página 194). `--no-split` suprime essa divisão da saída gerada.

  Alternativamente, `--split=`*como* pode ser usada para especificar em qual nível a saída HTML deveria ser dividida. Os valores possíveis para *como* são:

  ‘`chapter`’ A saída é dividida em `@chapter` e outros comandos @ de seccionamento nesse nível (`@appendix`, etc.).

  ‘`section`’ A saída é dividida em `@section` e similares.

  ‘`node`’ A saída é dividida em cada nó. Esse é o padrão.

  A saída de texto simples pode ser dividida similarmente ao HTML. Isso pode ser útil para extrair seções a partir de um documento do Texinfo e torná-las disponíveis como arquivos separados.

`--split-size=`*número*
: Mantém os arquivos Info com no máximo *número* caracteres, se possível; o padrão é 300.000. (Entretanto, um nó nunca será dividido entre os arquivos Info).

`--transliterate-file-names`
: Habilita transliteração de caracteres de 8 bits em nomes de nós para o propósito de criação de nomes de arquivo. Veja Seção 22.4.4 [Expansão de Caracteres de 8 bits do Xref do HTML], Página 200.

`-U` *variável*
: Causa *variável* ser indefinida. Isso é equivalente a `@clear` *variável* no arquivo do Texinfo (veja Seção 16.5 [`@set @clear @value`], Página 132).

`--verbose`
: Causa `makeinfo` exibir mensagens dizendo o que está fazendo. Normalmente, `makeinfo` somente emite mensagens se existirem erros ou avisos.

`--version`
`-V` Imprime o número da versão, então sai com sucesso.

`--Xopt` *string*
: Passa *string* (uma palavra do shell) para `texi2dvi`; pode ser repetida (veja Seção 20.3 [Saída Impressa do `texi2any`], Página 169).

`--xml` Gera saída XML do Texinfo (em vez de Info).

`makeinfo` também lê a variável de ambiente `TEXINFO_OUTPUT_FORMAT` para determinar o formato de saída, se não substituída por uma opção de linha de comando. O valor deveria ser um de:

```
docbook  dvi  dvipdf  html  info  pdf  plaintext  ps  xml
```

Se não configurada ou, de outra forma, especificada, a saída Info será o padrão.

A variável de personalização de mesmo nome também é lida; se configurada, isso substitui uma configuração de variável de ambiente, porém não uma opção de linha de comando. Veja Seção 20.5.2 [Variáveis e Opções de Personalização], Página 171.

20.3 Saída Impressa do `texi2any`

Para justificar o nome Texinfo-to-*any*, `texi2any` tem suporte básico para criar saída impressa nos vários formatos: DVI, PDF e PostScript, do TEX. Isso é feito pelo método simples de executar o programa `texi2dvi` quando esses formatos de saída são solicitados, depois de verificar a validade da entrada para dar para os(as) usuários(as) o benefício da verificação de erros do `texi2any`. Se você não quiser essa verificação de erros, talvez porque teu manual faça truques avançados do TEX junto com o `texinfo.tex`, basta invocar o `texi2dvi` diretamente.

As opções de formato de saída para isso são `--dvi`, `--dvipdf`, `--pdf` e `--ps`. Veja Seção 19.2 [Formatar com `texi2dvi`], Página 150, para mais detalhes acerca dessas opções e da operação geral do `texi2dvi`. Além disso, as opções `--verbose`, `--silent` e `--quiet` são passadas adiante se especificadas; as opções `-I` e `-o` são, da mesma maneira, passadas adiante com os argumentos delas, e `--debug` sem o argumento dela.

A única opção restante que está relacionada à invocação do `texi2dvi` é `--Xopt`. Aqui, apenas o argumento é passado adiante e várias opções `--Xopt` se acumulam. Isso fornece uma maneira de construir uma linha de comando arbitrária para `texi2dvi`. Por exemplo, executando

```
texi2any --Xopt -t --Xopt @a4paper --pdf foo.texi
```

é equivalente a executar

```
texi2dvi -t @a4paper --pdf foo.texi
```

exceto pela verificação de validade.

Embora alguém possa desejar que outras opções para `texi2any` tivessem efeito, elas não tem. Por exemplo, executar '`texi2any --no-number-sections --dvi foo.texi`' ainda resulta em um arquivo DVI com seções numeradas. (Talvez isso pudesse ser melhorado no futuro, se solicitações forem recebidas).

O nome real do comando que é invocado é especificado pela variável de personalização `TEXI2DVI` (veja Seção 20.5.4 [Outras Variáveis de Personalização], Página 177). Como você pode imaginar, o padrão é '`texi2dvi`'.

O próprio `texi2any` não gera nenhuma saída normal quando invoca o `texi2dvi`, somente mensagens de diagnóstico.

20.4 Validação de Ponteiro

Se você não suprimir a validação de ponteiro com a opção '`--no-validate`' ou com o comando `@novalidate` no arquivo fonte (veja Seção 19.1 [Use TEX], Página 150), `makeinfo` verificará a validade do arquivo do Texinfo.

A maioria das verificações de validação é diferente, dependendo de se os ponteiros de nó são determinados explicitamente ou implicitamente. Com ponteiros de nó explícitos, aqui está a lista do que é verificado:

1. Se uma referência de nó 'Próximo', 'Anterior' ou 'Acima' for uma referência para um nó no arquivo atual e não for uma referência externa, como para `(diretório)`, então o nó referenciado precisa existir.
2. Cada nó, exceto o nó 'Top', precisa ter um ponteiro 'Up'.
3. O nó referenciado por um ponteiro 'Acima' precisa ele próprio referenciar o nó atual por meio de um item de menu, a menos que o nó referenciado por 'Acima' tenha o formato '`(`*`arquivo`*`)`'.

Com ponteiros de nó implícitos, o erro acima não pode ocorrer, como tal. (Que é uma das principais razões pelas quais nós recomendamos usar esse recurso do `makeinfo`, e não especificar quaisquer ponteiros de nó você mesmo(a)).

Em vez disso, `makeinfo` verifica se a árvore construída a partir dos menus do documento corresponde à árvore construída a partir dos comandos de seccionamento. Por exemplo, se

um menu de nível de capítulo mencionar nós *n1* e *n2*, nessa ordem, nós *n1* e *n2* precisam ser associados com comandos `@section` no capítulo.

Finalmente, com ponteiros de nó explícitos e implícitos, `makeinfo` verifica se cada nó, exceto o nó 'Top', é referenciado em um menu.

20.5 Variáveis de Personalização

> **Aviso:** Esses nomes e significados de variáveis de personalização podem mudar em qualquer lançamento do Texinfo. Nós sempre tentamos evitar mudanças incompatíveis, mas não podemos prometer absolutamente, pois as necessidades mudam com o tempo.

Muitos aspectos do comportamento e da saída do `texi2any` podem ser modificados modificando as assim chamadas *variáveis de personalização*. Elas se enquadram em algumas categorias gerais:

- Aquelas associadas com comandos @; por exemplo, `@documentlanguage`.
- Aquelas associadas com opções de linha de comando; por exemplo, a variável de personalização `SPLIT` está associada com a opção de linha de comando `--split`, e `TEXINFO_OUTPUT_FORMAT` permite especificar o formato de saída.
- Aquelas associadas com personalização da saída HTML.
- Outras variáveis ad hoc.

Variáveis de personalização podem ser configuradas na linha de comando usando `--set-customization-variable '`*`variável valor`*`'` (aspando o par variável/valor para o shell) ou `--set-customization-variable` *`variável=valor`* (usando `=`). Um *valor* especial é '`undef`', que configura a variável para esse valor Perl especial "undefined".

As seções abaixo fornecem os detalhes para cada uma delas.

20.5.1 Variáveis de Personalização para Comandos @

Cada um dos seguintes comandos @ tem uma variável de personalização associada com o mesmo nome (menos o `@` inicial):

```
@allowcodebreaks @clickstyle @codequotebacktick
@codequoteundirected @contents @deftypefnnewline
@documentdescription @documentencoding @documentlanguage
@evenfooting   @evenfootingmarks
@evenheading   @evenheadingmarks
@everyfooting  @everyfootingmarks
@everyheading  @everyheadingmarks
@exampleindent @firstparagraphindent
@fonttextsize  @footnotestyle @frenchspacing @headings
@kbdinputstyle @novalidate
@oddfooting    @oddfootingmarks
@oddheading    @oddheadingmarks
@pagesizes     @paragraphindent
@setchapternewpage @setcontentsaftertitlepage
@setfilename   @setshortcontentsaftertitlepage @shortcontents
@urefbreakstyle @validatemenus @xrefautomaticsectiontitle
```

Configurar uma tal variável de personalização para um valor '`foo`' é semelhante a executar `@`*`comando`* `foo`. Não é exatamente o mesmo, no entanto, já que quaisquer efeitos colaterais da análise do fonte do Texinfo não são refeitos. Além disso, algumas variáveis não recebem código do Texinfo ao gerar formatos específicos, mas um argumento que já está formatado. Esse é o caso, por exemplo, para HTML para `documentdescription`.

20.5.2 Variáveis e Opções de Personalização

A tabela seguinte fornece as variáveis de personalização associadas com algumas opções de linha de comando. Veja Seção 20.2 [Invocando texi2any], Página 163, para o significado das opções.

Opção	Variável
--enable-encoding	ENABLE_ENCODING
--document-language	documentlanguage
--error-limit	ERROR_LIMIT
--fill-column	FILLCOLUMN
--footnote-style	footnotestyle
--force	FORCE
--internal-links	INTERNAL_LINKS
--macro-expand	MACRO_EXPAND
--headers	HEADERS, SHOW_MENU
--no-warn	NO_WARN
--no-validate	novalidate
--number-footnotes	NUMBER_FOOTNOTES
--number-sections	NUMBER_SECTIONS
--node-files	NODE_FILES
--output	OUT, OUTFILE, SUBDIR
--paragraph-indent	paragraphindent
--silent	SILENT
--split	SPLIT
--split-size	SPLIT_SIZE
--transliterate-file-names	TRANSLITERATE_FILE_NAMES
--verbose	VERBOSE

Configurar uma tal variável de personalização para um valor ‘foo’ é essencialmente o mesmo que especificar o *--opção*=foo se a opção receber um argumento, ou *--opção* se não.

Além disso, a variável de personalização TEXINFO_OUTPUT_FORMAT permite especificar o que makeinfo produz, seja um dos formatos usuais de saída que podem ser especificados com opções, ou vários outros formatos:

‘docbook’
‘dvi’
‘dvipdf’
‘html’
‘info’
‘pdf’
‘plaintext’
‘ps’
‘xml’ Esses correspondem às opções de linha de comando (e valores da variável de ambiente TEXINFO_OUTPUT_FORMAT) do mesmo nome. Veja Seção 20.2 [Invocando texi2any], Página 163.

‘debugcount’
Em vez de gerar um formato regular de saída, emite a contagem de bytes e de linhas obtidas ao converter para Info e outras informações.

‘debugtree’
Em vez de gerar um formato regular de saída, produz uma representação de texto da árvore obtida pela análise do documento de entrada do Texinfo.

‘parse’ Faz somente análise do fonte do Texinfo; não existe saída gerada.

‘plaintexinfo’
: Produz o fonte do Texinfo com todas as macros, @include e @value{} expandidas. Isso é similar a configurar --macro-expand, mas em vez de ser produzido em adição à conversão normal, a saída de Texinfo é a saída principal.

‘rawtext’
: Produz texto bruto, com formatação mínima. Por exemplo, notas de rodapé são ignoradas e não existe preenchimento de parágrafo. Isso é usado pelo analisador para nomes de arquivo e texto de direitos de cópia em comentários HTML, por exemplo.

‘structure’
: Faz somente análise do fonte do Texinfo e determinação da estrutura do documento; não existe saída gerada.

‘texinfosxml’
: Produz o documento na representação TexinfoSXML, uma sintaxe para escrever dados XML usando expressões S da Lisp.

‘textcontent’
: Produz somente o conteúdo do texto, despojado de comandos; isso é útil para verificação ortográfica ou contagem de palavras, por exemplo. A script trivial detexinfo configurando isso está no diretório util do fonte do Texinfo como um exemplo. É uma linha:

```
exec texi2any -c TEXINPUT_OUTPUT_FORMAT=conteúdotexto "$@"
```

20.5.3 Variáveis de Personalização de HTML

Essa tabela fornece as variáveis de personalização que se aplicam somente à saída HTML. Algumas outras variáveis de personalização se aplicam tanto ao HTML quanto a outros formatos de saída; essas são fornecidas na próxima seção.

AVOID_MENU_REDUNDANCY
: Para HTML. Se configurada, e a entrada do menu e a descrição do menu forem as mesmas, então não imprime a descrição do menu; padrão falso.

AFTER_BODY_OPEN
: Para HTML. Se configurada, o texto correspondente aparecerá no início de cada arquivo HTML; padrão desconfigurada.

AFTER_ABOUT
: Para HTML, quando um elemento About é produzido. Se configurada, o texto correspondente aparecerá no final do elemento About; padrão desconfigurada.

AFTER_OVERVIEW
AFTER_TOC_LINES
: Para HTML. Se configurada, o texto correspondente é produzido depois do tabela curta de conteúdo para AFTER_OVERVIEW e depois do sumário para AFTER_TOC_LINES; caso contrário, uma string padrão é usada. No momento da escrita, um elemento é fechado.

  Em geral, você deveria configurar BEFORE_OVERVIEW se AFTER_OVERVIEW estiver configurada, e deveria configurar BEFORE_TOC_LINES se AFTER_TOC_LINES estiver configurada.

BASEFILENAME_LENGTH
: Para HTML. O comprimento máximo dos nomes de arquivo base; padrão 245. Mudar isso tornaria referências manuais cruzadas para esses nomes longos de nó inválidas (veja Seção 22.4.1 [Fundamentos do Link Xref do HTML], Página 197).

`BEFORE_OVERVIEW`
`BEFORE_TOC_LINES`
: Para HTML. Se configurada, o texto correspondente é produzido antes do tabela curta de conteúdo para `BEFORE_OVERVIEW` e antes do sumário para `BEFORE_TOC_LINES`, caso contrário, uma string padrão é usada. No momento da escrita, um elemento `<div ...>` é aberto.

  Em geral, você deveria configurar `AFTER_OVERVIEW` se `BEFORE_OVERVIEW` estiver configurada, e deveria configurar `AFTER_TOC_LINES` se `BEFORE_TOC_LINES` estiver configurada.

`BIG_RULE` Para HTML. Regra usada depois e antes do elemento superior e antes de elementos especiais, mas não para rodapés e cabeçalhos; padrão `<hr>`.

`BODYTEXT` Para HTML, o texto que aparece em `<body>`. Por padrão, configura o atributo `lang` do HTML para o idioma do documento (veja Seção 15.1 [`@documentlanguage`], Página 125).

`CASE_INSENSITIVE_FILENAMES`
: Para HTML. Constrói nomes de arquivos de saída como se o sistema de arquivos não diferenciasse maiúsculas de minúsculas (veja Seção 22.2 [Divisão de HTML], Página 195); padrão falso.

`CHAPTER_HEADER_LEVEL`
: Para HTML. Nível de formatação de cabeçalho usado para comandos de seccionamento de nível de capítulo; padrão '`2`'.

`CHECK_HTMLXREF`
: Para HTML. Verifica se os manuais que são alvo de referências cruzadas externas (veja Seção 6.4.4 [Quatro e Cinco Argumentos], Página 48) estão presentes em `htmlxref.cnf` (veja Seção 22.4.6 [Configuração do Xref do HTML], Página 201); padrão falso.

`COMPLEX_FORMAT_IN_TABLE`
: Para HTML. Se configurado, usa tabelas para recuo de formatos complexos; padrão falso.

`CSS_LINES`
: Para HTML. Saída CSS, determinada automaticamente por padrão (veja Seção 22.3 [CSS de HTML], Página 195).

`DATE_IN_HEADER`
: Para HTML. Coloca a data de geração do documento no cabeçalho; desligada por padrão.

`DEF_TABLE`
: Para HTML. Se configurada, uma construção `<table>` para `@deffn` e comandos @ semelhantes é usada (parecendo mais com a saída do TEX), em vez de listas de definições; padrão falso.

`DEFAULT_RULE`
: Para HTML. Regra usada entre elementos, exceto antes e depois do elemento superior, e antes de elementos especiais, e para rodapés e cabeçalhos; padrão `<hr>`.

`DO_ABOUT` Para HTML. Se configurada para 0, nunca faz um elemento especial About; se configurada para 1, sempre faz um elemento especial About; padrão 0.

`EXTERNAL_DIR`
: Para HTML. Diretório base para manuais externos; padrão nenhum. É melhor usar o mecanismo geral de referência cruzada externa (veja Seção 22.4.6 [Configuração do Xref do HTML], Página 201) que essa variável.

EXTRA_HEAD
: Para HTML. Texto adicional aparecendo dentro de <head>; padrão desconfigurada.

FOOTNOTE_END_HEADER_LEVEL
: Para HTML. Nível de formatação de cabeçalho usado para o cabeçalho de notas de rodapé com o estilo de notas de rodapé 'end'; padrão '4'. Veja Seção 10.3.2 [Estilos de Notas de Rodapé], Página 87.

FOOTNOTE_SEPARATE_HEADER_LEVEL
: Para HTML. Nível de formatação de cabeçalho usado para o cabeçalho de notas de rodapé com o estilo de notas de rodapé 'separate'; padrão '4'. Veja Seção 10.3.2 [Estilos de Notas de Rodapé], Página 87.

FRAMES
: Para HTML. Se configurada, um arquivo descrevendo o esquema do quadro é gerado, junto com um arquivo com a tabela curta de conteúdo; padrão falso.

FRAMESET_DOCTYPE
: Para HTML. O mesmo que DOCTYPE, mas para o arquivo contendo a descrição do quadro.

HEADER_IN_TABLE
: Para HTML. Usa tabelas para formatação de cabeçalho em vez de um simples elemento ; padrão falso.

ICONS
: Para HTML. Usa ícones para o painel de navegação; padrão falso.

IMAGE_LINK_PREFIX
: Para HTML. Se configurada, o valor associado é anteposto nos links do arquivo de imagem; padrão desconfigurada.

INLINE_CONTENTS
: Para HTML. Se configurada, produz o conteúdo onde @contents e comandos @- similares estão localizados; padrão verdadeiro. Isso é ignorado se @set*contentsaftertitlepage estiver configurado (veja Seção 3.5 [Conteúdo], Página 22).

INLINE_CSS_STYLE
: Para HTML. Coloca CSS diretamente em elementos HTML em vez de no início da saída; padrão falso.

KEEP_TOP_EXTERNAL_REF
: Para HTML. Se configurada, não ignora 'Top' como o primeiro argumento para uma referência externa a um manual, como é feito por padrão. Veja Seção 6.5 [Referenciando Um Manual Como Um Todo], Página 49.

L2H
: Para HTML. Se configurada, latex2html é usado para converter seções @math e @tex; padrão falso. Melhor usada com --iftex.

L2H_CLEAN
: (Relevante somente se L2H estiver configurada). Se configurada, os arquivos intermediários gerados em relação ao latex2html serão removidos; padrão verdadeiro.

L2H_FILE
: (Relevante somente se L2H estiver configurada). Se configurada, o arquivo fornecido será usado como arquivo de iniciação do latex2html; padrão desconfigurada.

L2H_HTML_VERSION
: (Relevante somente se L2H estiver configurada). A versão HTML usada na chamada latex2html; padrão desconfigurada.

L2H_L2H
: (Relevante somente se L2H estiver configurada). O programa invocado como latex2html; padrão é latex2html.

`L2H_SKIP` (Relevante somente se `L2H` estiver configurada). Se configurada para um valor verdadeiro, a chamada real para `latex2html` é ignorada; o conteúdo gerado anteriormente é reusado. Se configurada para 0, o cache não é usado. Se configurada para '`undef`', o cache é usado para tantos fragmentos do TEX quanto possível e para qualquer restante o comando é executado. O padrão é '`undef`'.

`L2H_TMP` (Relevante somente se `L2H` estiver configurada). Configura o diretório usado para arquivos temporários. Nenhum dos componentes de nome de arquivo nesse nome de diretório pode começar com '`.`'; caso contrário, `latex2html` falhará (por causa de `dvips`). O padrão é a string vazia, o que significa o diretório atual.

`MAX_HEADER_LEVEL`
Para HTML. Nível máximo de formatação de cabeçalho usado (números de nível de formatação de cabeçalho mais altos correspondem a níveis de seccionamento mais baixos); padrão '`4`'.

`MENU_SYMBOL`
Para HTML. Símbolo usado na frente de entradas de menu quando nomes de nós são usados para formatação de entradas de menu; padrão '`&bull;`'.

`MONOLITHIC`
Para HTML. Produz somente um arquivo incluindo o sumário. Configurada por padrão, mas relevante somente quando a saída não for dividida.

`NO_CSS` Para HTML. Não usa CSS; padrão falso. Veja Seção 22.3 [CSS de HTML], Página 195.

`NODE_FILE_EXTENSION`
Para HTML. Extensão para arquivos de nó se `NODE_FILENAMES` estiver configurada; padrão '`html`'.

`PRE_ABOUT`
Para HTML, quando um elemento About é produzido. Se configurada para uma string de texto, esse texto aparecerá no início do elemento About. Se configurada para uma referência sobre uma sub rotina, o resultado da chamada da suberitínea aparecerá no início do elemento About. Se não configurada (o padrão), o texto padrão é usado.

`PRE_BODY_CLOSE`
Para HTML. Se configurada, o texto fornecido aparecerá no rodapé de cada arquivo HTML; padrão desconfigurada.

`PROGRAM_NAME_IN_FOOTER`
Para HTML. Se configurada, produz o nome do programa e informações diversas relacionadas nos rodapés da página; padrão falso.

`SHORTEXTN`
Para HTML. Se configurada, usa '`.htm`' como extensão; padrão falso.

`SHOW_TITLE`
Para HTML. Se configurada, produz o título no início do documento; padrão verdadeiro.

`SIMPLE_MENU`
Para HTML. Se configurada, usa um estilo pré-formatado simples para o menu, em vez de dividir as diferentes partes do menu; padrão falso. Veja Seção 4.9.4 [Partes de Menu], Página 37.

`TOC_LINKS`
: Para HTML. Se configurada, links oriundos de títulos para entradas da tabela de conteúdo são criados; padrão falso.

`TOP_FILE`
: Esse nome de arquivo pode ser usado para o arquivo de nível superior. A extensão é configurada apropriadamente, se necessário. Isso é usado para substituir o padrão e, em geral, é levado em conta somente quando a saída é dividida e para HTML.

`TOP_NODE_FILE`
: Para HTML. Nome do arquivo usado para o nó Top, se `NODE_FILENAMES` estiver configurada; padrão é `index`.

`TOP_NODE_FILE_TARGET`
: Para HTML. Nome do arquivo usado para o nó Top em referências cruzadas; padrão é `index`.

`TOP_NODE_UP_URL`
: Para HTML. Uma URL usada para referências `(dir)`; o padrão é `undef`, significando que as regras normais se aplicam, tipicamente levando a um link para '`dir.html`' originário de uma referência implícita ou explícita para '`(dir)`' (veja Seção 22.4 [Xref de HTML], Página 197). Para mais acerca dos ponteiros do nó Top, veja Seção 4.5 [Primeiro Nó], Página 32. Para substituir o ponteiro Acima em outros formatos, veja-se `TOP_NODE_UP` em Seção 20.5.4 [Outras Variáveis de Personalização], Página 177.

`USE_ACCESSKEY`
: Para HTML. Usa `accesskey` em referências cruzadas; padrão verdadeiro.

`USE_ISO`
: Para HTML. Usa entidades para caracteres duplicados de aspas simples (veja Seção 12.5 [Inserindo Aspas], Página 101), e '`---`' e '`--`' (veja Seção 2.1 [Convenções], Página 9); padrão verdadeiro.

`USE_LINKS`
: Para HTML. Gera elementos `<link>` na saída `<head>` do HTML; padrão verdadeiro.

`USE_REL_REV`
: Para HTML. Usa `rel` em referências cruzadas; padrão verdadeiro.

`VERTICAL_HEAD_NAVIGATION`
: Para HTML. Se configurada, um painel vertical de navegação é usado; padrão falso.

`WORDS_IN_PAGE`
: Para HTML, com saída dividida em nós. Especifica o comprimento mínimo aproximado da página em que um painel de navegação é colocado na parte inferior de uma página. Para evitar sempre ter os botões de navegação na parte inferior de uma página, configure isso para um número suficientemente grande. O padrão é 300.

`XREF_USE_FLOAT_LABEL`
: Para HTML. Se configurada, para o nome do flutuador em referências cruzadas, usa o rótulo do flutuador em vez do tipo seguido pelo número do flutuador (veja Seção 10.1.1 [`@float`], Página 82). O padrão é desligado.

`XREF_USE_NODE_NAME_ARG`
: Para HTML. Relevante somente para comandos de referência cruzada sem nome de referência cruzada (segundo argumento). Se configurada para 1, usa o argumento do nome do nó (primeiro) em comandos @ de referência cruzada para o texto exibido como o hiperlink. Se configurada para 0, usa o nome do nó se `USE_NODES` estiver configurado, caso contrário, o nome da seção. Se configurada para '`undef`', usa o primeiro argumento em ambientes pré-formatados, caso contrário, usa o nome do nó ou o nome da seção, dependendo de `USE_NODES`. O padrão é '`undef`'.

20.5.4 Outras Variáveis de Personalização

Esta tabela fornece as restantes variáveis de personalização, as quais se aplicam a vários formatos, afetam o comportamento global ou não se enquadram nas categorias das seções anteriores.

`CLOSE_QUOTE_SYMBOL`
: Quando uma aspa de fechamento for necessária, usa esse caractere; padrão `&rsquo;` em HTML, `&#8217;` em Docbook. O padrão para Info é o mesmo que `OPEN_QUOTE_SYMBOL` (veja-se abaixo).

`CPP_LINE_DIRECTIVES`
: Reconhecer diretivas `#line` em uma passagem de "pré-processamento" (veja Seção 17.6 [Processadores Externos de Macro], Página 143); ligada por padrão.

`DEBUG` Se configurada, a saída de depuração é gerada; padrão é desligada (zero).

`DOCTYPE` Para Docbook, HTML, XML. Especifica o `SystemLiteral`, o identificador de sistema da entidade. Esse é um URI que pode ser usado para recuperar a entidade e identifica o DTD canônico para o documento. O valor padrão é diferente para cada HTML, Docbook e Texinfo XML.

`DUMP_TEXI`
: Para depuração. Se configurada, nenhuma conversão é feita, somente análise e expansão de macro. Se a opção `--macro-expand` for configurada, o fonte do Texinfo também será expandido para o arquivo correspondente. Padrão falso.

`DUMP_TREE`
: Para depuração. Se configurada, a árvore construída ao analisar um documento do Texinfo é enviada para erro padrão; padrão falso.

`ENABLE_ENCODING_USE_ENTITY`
: Para HTML, XML. Se `--enable-encoding` estiver configurada, e existir uma entidade correspondendo com a letra ou o símbolo sendo produzido, prefira a entidade. Configurada por padrão para HTML, mas não para XML.

`EXTERNAL_CROSSREF_SPLIT`
: Para referências cruzadas para outros manuais, isso determina se o outro manual é considerado dividido ou monolítico. Por padrão, é configurada baseada no valor de `SPLIT`. Veja Seção 22.4 [Xref de HTML], Página 197, e veja Seção 22.4.6 [Configuração do Xref do HTML], Página 201.

`EXTENSION`
: A extensão adicionada ao nome do arquivo de saída. O padrão é diferente para cada formato de saída.

`FIX_TEXINFO`
: Para "Texinfo simples" (veja-se o item `PLAINTEXINFO`). Se configurada para falso, o Texinfo resultante não terá todos os erros corrigidos, tal como '`@end`' ausente; padrão verdadeiro. Essa variável somente é relevante ao expandir Texinfo; outros conversores sempre tentam produzir algo sensato, mesmo que a entrada seja errônea.

`IGNORE_BEFORE_SETFILENAME`
: Se configurada, inicia a saída em `@setfilename`, se `@setfilename` estiver presente; padrão verdadeiro.

`IGNORE_SPACE_AFTER_BRACED_COMMAND_NAME`
: Se configurado, espaços são ignorados depois de um comando @ que receba chaves. Padrão verdadeiro, correspondendo ao comportamento do TEX.

`INDEX_ENTRY_COLON`
: Símbolo usado entre a entrada do índice e o nó ou seção associado; padrão '`:`'.

`INDEX_SPECIAL_CHARS_WARNING`
: Se configurada, avisa acerca de '`:`' na entrada de índice, pois isso leva a entradas inválidas em menus de índice em arquivos de saída do Info. Somente para Info e texto simples.

`INFO_SPECIAL_CHARS_QUOTE`
: Se configurada, sempre que existir caracteres problemáticos para a saída Info em locais como nomes de nós ou itens de menu, envolva entre aspas a parte da construção onde eles aparecem, conforme descrito em Apêndice G [Especificação do Formato Info], Página 259. Veja Seção 4.4 [Exigências de Linha de Nó], Página 31.

`INFO_SPECIAL_CHARS_WARNING`
: Se configurada, avisa acerca de construções problemáticas para saída Info (como a string '`::`') em nomes de nós, itens de menu e referências cruzadas; padrão verdadeiro. Não avisa acerca de entradas de índice, já que problemas de análise não impedem a navegação; leitores ainda conseguem relativamente facilmente encontrar o caminho para o nó em questão.

`INLINE_INSERTCOPYING`
: Se configurada, `@insertcopying` será substituído pelo conteúdo de `@copying` (veja Seção 3.3.1 [`@copying`], Página 17) como se `@insertcopying` fosse uma macro definida por usuário(a); padrão falso.

`INPUT_ENCODING_NAME`
: Nome de codificação normalizado adequado para saída. Deveria ser um nome de conjunto de caracteres usável em HTML, tipicamente um dos nomes de codificação preferidos da IANA. Você não deveria precisar usar essa variável, pois ela é configurada por `@documentencoding` (veja Seção 15.2 [`@documentencoding`], Página 126).

`INPUT_PERL_ENCODING`
: Codificação Perl usada para processar o fonte do Texinfo. Você não deveria precisar usar essa variável, pois ela é configurada por `@documentencoding` (veja Seção 15.2 [`@documentencoding`], Página 126).

`MACRO_BODY_IGNORES_LEADING_SPACE`
: Ignora espaço em branco no início da linha do corpo da macro definida por usuário(a), imitando uma limitação do TEX (veja Seção 17.3 [Detalhes de Macro], Página 140). Padrão desligado.

`MAX_MACRO_CALL_NESTING`
: O número máximo de chamadas recursivas de comandos @ definidos por meio de `@rmacro`; padrão 100000. O propósito dessa variável é o de evitar recursões infinitas.

`MENU_ENTRY_COLON`
: Símbolo usado entre a entrada do menu e a descrição; padrão '`:`'.

`NO_USE_SETFILENAME`
: Se configurada, não use `@setfilename` para configurar o nome do documento; em vez disso, baseie o nome do documento de saída somente no nome do arquivo de entrada. O padrão é falso.

`NODE_FILENAMES`
: Se configurada, nomes de nó são usados para construir nomes de arquivo. Por padrão, é configurada se a saída for dividida por nó, ou se `NODE_FILES` estiver configurada e a saída for dividida de alguma forma.

`NODE_NAME_IN_INDEX`
: Se configurada, use nomes de nós em entradas de índice, caso contrário, prefira nomes de seção; padrão verdadeiro.

`NODE_NAME_IN_MENU`
: Se configurada, use nomes de nós em entradas de menu, caso contrário, prefira nomes de seção; padrão verdadeiro.

`OPEN_QUOTE_SYMBOL`
: Quando uma aspa de abertura for necessária, por exemplo, para a saída do '`@samp`', use o caractere especificado; padrão `&lsquo;` para HTML, `&#8216;` para Docbook. Para Info, o padrão depende da codificação habilitada do documento (veja Seção 15.2 [`@documentencoding`], Página 126); se nenhuma codificação de documento estiver configurada, ou a codificação for US-ASCII, etc., '`'`' será usado. Esse caractere geralmente aparece como uma aspa simples não direcionada em sistemas modernos. Se a codificação do documento for Unicode, a saída Info usará uma aspa esquerda Unicode.

`OUTPUT_ENCODING_NAME`
: Nome de codificação normalizado usado para arquivos de saída. Deveria ser um nome de conjunto de caracteres usável em HTML, tipicamente um dos nomes de codificação preferidos da IANA. Por padrão, se uma codificação de entrada for configurada (tipicamente por meio de `@documentencoding` ou `INPUT_ENCODING_NAME`), essa informação será usada para configurar o nome da codificação de saída. Se nenhuma codificação de entrada for especificada, o nome padrão da codificação de saída poderá ser configurado pelo formato de saída. Em particular, os formatos baseados em XML usam `utf-8` para `OUTPUT_ENCODING_NAME` se a codificação não for especificada de outra maneira. Veja Seção 15.2 [`@documentencoding`], Página 126.

`OVERVIEW_LINK_TO_TOC`
: Se configurada, as referências cruzadas na Visão Geral se lincam às entradas correspondentes do Sumário; padrão verdadeiro.

`PACKAGE`
`PACKAGE_VERSION`
`PACKAGE_AND_VERSION`
`PACKAGE_URL`
`PACKAGE_NAME`
: O nome curto do pacote de implementação, versão do pacote, nome do pacote e versão concatenados, url do pacote e nome completo do pacote, respectivamente. Por padrão, essas variáveis são todas configuradas por meio do Autoconf, Automake e `configure`.

`PREFIX` O prefixo do arquivo de saída, o qual é anteposto a alguns nomes de arquivo de saída. Por padrão, ele é configurado por `@setfilename` ou a partir do arquivo de entrada (veja Seção 3.2.3 [`@setfilename`], Página 16). Como esse valor é usado depende do valor de outras variáveis de personalização ou de opções de linha de comando, como se a saída estivesse dividida, e de `NODE_FILENAMES`. O padrão é desconfigurada.

`PROGRAM` Nome do programa usado. Por padrão, é configurada para o nome do programa iniciado, com um '`.pl`' final removido.

`RENAMED_NODES_FILE`
: Se configurada, use o valor para o arquivo de descrição de nós renomeados. Se não configurada, o arquivo é *`nome_base_documento`*`-noderename.cnf`. Veja Seção 22.4.7 [Preservação de Link Xref do HTML], Página 203.

`RENAMED_NODES_REDIRECTIONS`
: Se configurada, cria arquivos de redirecionamento para nós renomeados. Configurada por padrão ao gerar HTML.

`SHOW_MENU`
: Se configurada, os menus do Texinfo são emitidos. Por padrão, é configurada, a menos que gerando Docbook ou se `--no-headers` for especificada.

`SORT_ELEMENT_COUNT`
: Se configurada, o nome de um arquivo para o qual uma lista de elementos (nós ou seções, dependendo do formato de saída) é despejada, ordenada pelo número de linhas que eles contém depois da remoção de comandos @; padrão desconfigurada. Isso é usado pelo programa `texi-elements-by-size` no diretório `util/` da distribuição do fonte do Texinfo (veja [texi-elements-by-size], Página 230).

`SORT_ELEMENT_COUNT_WORDS`
: Ao despejar o arquivo de elementos por tamanho (veja-se o item precedente), use contagens de palavras em vez de contagens de linhas; padrão falso.

`TEST`
: Se configurada para verdadeiro, algumas variáveis que normalmente são geradas dinamicamente novamente para cada execução (data, nome do programa, versão) são configuradas para valores fixos e fornecidos. Isso é útil para comparar a saída com um arquivo de referência, como é feito para os testes. O padrão é falso.

`TEXI2DVI`
: Nome do comando usado para produzir PostScript, PDF e DVI; padrão '`texi2dvi`'. Veja Seção 20.3 [Saída Impressa do `texi2any`], Página 169.

`TEXI2HTML`
: Gere HTML e tente ser o mais compatível possível com `texi2html`; padrão falso.

`TEXINFO_COLUMN_FOR_DESCRIPTION`
: Usado com a transformação de árvore do `indent_menu_descriptions`, descrita abaixo; padrão 32 (correspondente a `texinfo-column-for-description` no Emacs)).

`TEXINFO_DTD_VERSION`
: Para XML. Versão do DTD usado no preâmbulo de saída XML. O padrão é configurada baseada em uma variável em `configure.ac`.

`TEXTCONTENT_COMMENT`
: Para saída de conteúdo de texto despojado (por exemplo, quando `TEXINFO_OUTPUT_FORMAT` for configurada para `textcontent`). Se configurada, também emite comentários. Padrão falso.

`TOP_NODE_UP`
: Nó superior para o nó Top; padrão '`(dir)`'. Para substituir a URL na saída HTML, veja-se `TOP_NODE_UP_URL` em Seção 20.5.3 [Variáveis de Personalização de HTML], Página 172.

`TREE_TRANSFORMATIONS`
: O valor associado é uma lista separada por vírgulas de transformações que podem ser aplicadas à árvore do Texinfo antes de emitir o resultado. Se mais que uma for especificada, a ordenamento é irrelevante; cada uma é sempre aplicada no ponto necessário durante o processamento.

  O único executado por padrão é '`move_index_entries_after_items`' para saída HTML e Docbook. Aqui está um exemplo de atualização do menu mestre em um documento:

```
makeinfo \
  -c TREE_TRANSFORMATIONS=regenerate_master_menu \
  -c PLAINTEXINFO=1 \
  meudocumento.texi \
```

```
-o /tmp/saida
```

(Ressalva: como a saída de PLAINTEXINFO expande macros e condicionais do Texinfo, é necessário remover quaisquer diferenças antes de instalar as atualizações no documento original. Isso será remediado em um lançamento futuro).

As seguintes transformações são atualmente suportadas (muitas são usadas no utilitário pod2texi distribuído com o Texinfo; veja Seção 20.7 [Invocando pod2texi], Página 183):

'complete_tree_nodes_menus'
: Adicione entradas de menu ou menus inteiros para nós associados com seções de qualquer nível, baseados na árvore de seccionamento.

'fill_gaps_in_sectioning'
: Adiciona seções @unnumbered... vazias em uma árvore para preencher lacunas no seccionamento. Por exemplo, um @unnumberedsec será inserido se um @chapter for seguido por um @subsection.

'indent_menu_descriptions'
: Reformate os menus de forma que as descrições comecem na coluna TEXINFO_COLUMN_DESCRIPTION.

'insert_nodes_for_sectioning_commands'
: Insira nós para comandos de seccionamento carecendo de um nó correspondente.

'move_index_entries_after_items'
: Em @enumerate e @itemize, mova as entradas de índice que aparecem logo antes de um @item para logo depois do @item. As linhas de comentários entre as entradas de índice também são movidas. Como mencionado, isso é sempre feito para saídas em HTML e em Docbook.

'regenerate_master_menu'
: Atualize o menu mestre do nó Top, ou substituindo o (primeiro) @detailmenu no menu do nó Top ou criando-o no final do menu do nó Top.

'simple_menu'
: Basicamente o mesmo que SIMPLE_MENU: use um estilo simples préformatado para o menu. Ele difere da configuração de SIMPLE_MENU em que SIMPLE_MENU somente tem efeito em saída HTML.

USE_NODES
: Use preferencialmente nós para decidir onde os elementos são separados. Se configurada para falso, use preferencialmente seccionamento para decidir onde os elementos são separados. O padrão é verdadeiro.

USE_NODE_TARGET
: Se configurada, use o nó associado com uma seção para o alvo de seção em referências cruzadas; padrão verdadeiro.

USE_NUMERIC_ENTITY
: Para HTML e XML. Se configurada, use entidades numéricas em vez de caracteres ASCII quando não existir entidade nomeada. Por padrão, configurada para verdadeiro para HTML.

USE_UP_NODE_FOR_ELEMENT_UP
: Preencha a direção de seccionamento para cima com a direção do nó quando não existir direção de seccionamento para cima. Na prática, isso somente pode acontecer quando não existir seção @top. Não configurada por padrão.

`USE_SETFILENAME_EXTENSION`
: O padrão é on para Info, off para outras saídas. Se configurada, use exatamente o que `@setfilename` fornecer para o nome do arquivo de saída, incluindo a extensão. Você não deveria precisar configurar explicitamente essa variável.

`USE_TITLEPAGE_FOR_TITLE`
: Use o `@titlepage` completo como título, não uma simples string de título; padrão falso.

`USE_UNIDECODE`
: Se configurada para falso, não use o módulo `Text::Unidecode` do Perl para transliterar mais caracteres; padrão verdadeiro.

20.6 Internacionalização de Strings de Documentos

`texi2any` escreve strings fixas no documento de saída em vários lugares: referências cruzadas, rodapés de página, a página de ajuda, texto alternativo para imagens e assim por diante. A string escolhida depende do valor do `documentlanguage` ao tempo em que a string sendo gerada (veja Seção 15.1 [`@documentlanguage`], Página 125, para a interface de comando do Texinfo).

O framework Gettext é usado para essas strings (veja *Gettext*). O pacote `libintl-perl` é usado como a implementação `gettext`; mais especificamente, a implementação Perl pura é usada, de forma que o Texinfo pode suportar comportamento consistente em todas as plataformas e instalações, o que não seria possível de outra maneira. `libintl-perl` é incluído na distribuição do Texinfo e sempre instalado, para garantir que esteja disponível se necessário. Também é possível usar o `gettext` do sistema (a escolha pode ser feita em tempo de construção).

O domínio Gettext '`texinfo_document`' é usado para as strings. Strings traduzidas são escritas como Texinfo e podem incluir comandos @. Em strings traduzidas, as partes variáveis da string geralmente não são denotadas por `%s` e similares, mas por '`{arg_name}`'. (Essa convenção é comum para `gettext` em Perl e é totalmente suportada no GNU Gettext; veja Seção "Perl Format Strings" em *GNU Gettext*). Por exemplo, no seguinte, '`{section}`' será substituído pelo nome da seção:

```
veja-se {section}
```

Essas strings de formato de chaves no estilo Perl são usadas por dois motivos: primeiro, a mudança da ordem dos argumentos de `printf` somente está disponível desde o Perl 5.8.0; segundo, e mais importante, a ordem dos argumentos é imprevisível, já que a expansão do comando @ possivelmente leve a ordens diferentes dependendo do formato de saída.

A expansão de uma string de tradução é feita assim:

1. Primeiro, a string é traduzida. A localidade é *@documentlanguage*`.`*@documentencoding*.

   Se o *@documentlanguage* tiver o formato '`ll_CC`', esse será tentado primeiro, e então apenas '`ll`'. Se esse não existir, e a codificação não for `us-ascii`, então `us-ascii` será tentado.

   A ideia é a de que se existir uma codificação `us-ascii`, isso significa que todos os caracteres no conjunto de caracteres podem ser expressos como comandos @. Por exemplo, existe uma localidade, `fr.us-ascii`, que pode acomodar qualquer codificação, já que todos os caracteres Latin 1 tem comandos @ associados. Por outro lado, o japonês tem somente uma tradução, `ja.utf-8`, já que não existem comandos @ para caracteres japoneses.
2. Em seguida, a string é expandida como Texinfo e convertida. Os argumentos são substituídos; por exemplo, '`{arg_name}`' é substituído pelo argumento real correspondente.

No exemplo seguinte, '`{date}`', '`{program_homepage}`' e '`{program}`' são os argumentos da string. Como eles são usados em `@uref`, a ordem deles não é previsível. '`{date}`', '`{program_homepage}`' e '`{program}`' são substituídos depois da expansão:

```
Gerado em @emph{{date}} usando
```

```
@uref{{program_homepage}, @emph{{program}}}.
```

Essa abordagem é reconhecidamente um pouco complicada. A utilidade dela é a de que ela suporta ter traduções disponíveis em diferentes codificações para codificações que podem ser cobertas por comandos @, e também especifica como a formatação para alguns comandos é feita, independentemente do formato de saída—mas ainda ser dependente do idioma. Por exemplo, a string de tradução '`@pxref`' pode ser assim:

```
veja-se {node_file_href} seção `{section}\' em @cite{{book}}
```

que permite especificar uma string independentemente do formato de saída, embora, com formatação avançada, ela possa ser traduzida adequadamente em muitos idiomas.

20.7 Invocando `pod2texi`: Converte POD para Texinfo

O programa `pod2texi` traduz arquivo(s) de documentação do pod do Perl para Texinfo. Existem dois modos básicos de operação: gerar um manual independente a partir de cada pod de entrada ou (se `--base-level=1` ou superior for fornecido) gerar subarquivos do Texinfo adequados para uso com `@include`.

Embora normalmente essa documentação no manual do Texinfo seja o melhor lugar para procurar, nesse caso nós documentamos todas as opções e exemplos no próprio programa `pod2texi`, já que ele pode ser útil fora do resto do Texinfo. Portanto, por favor, veja-se a saída do `pod2texi --help`, a versão na web em `http://www.gnu.org/software/texinfo/manual/pod2texi.html`, etc.

Para um exemplo de uso do `pod2texi` para criar o Texinfo a partir da própria documentação do Perl, veja-se `contrib/perldoc-all` (`http://svn.savannah.gnu.org/viewvc/trunk/contrib/perldoc-all/?root=texinfo`) na distribuição do fonte do Texinfo (a saída está disponível em `http://www.gnu.org/software/perl/manual`).

20.8 `texi2html`: Ancestral de `texi2any`

Conceitualmente, o programa `texi2html` é o ascendente do programa `texi2any` de hoje. O `texi2html` foi desenvolvido independentemente, originalmente por Lionel Cons em 1998; na época, o `makeinfo` não conseguia gerar HTML. Muitas outras pessoas contribuíram para o `texi2html` ao longo dos anos.

O `texi2any` atual usa pouco do código real do `texi2html` e tem uma abordagem básica bem diferente para a implementação (ou seja, analisar o documento do Texinfo em uma árvore), mas ainda assim, existe uma semelhança familiar.

Por projeto, `texi2any` suporta quase todos os recursos de `texi2html` de alguma forma. No entanto, nós não tentamos manter compatibilidade estrita, de forma que nenhum executável `texi2html` é instalado pelo pacote do Texinfo. Uma aproximação pode ser executada com uma invocação como esta (disponível como `util/texi2html` no fonte do Texinfo):

```
texi2any --set-customization-variable TEXI2HTML=1 ...
```

mas, para enfatizar, isso *não* é um substituto imediato para o `texi2html` anterior. Aqui estão as maiores diferenças:

- Mais flagrantemente, as opções de linha de comando do `texi2html` agora são variáveis de personalização, na maior parte. Uma tabela de equivalentes aproximados é fornecida abaixo.
- A API de personalização em nível de programa é muito diferente no `texi2any`.
- Índices não podem ser divididos.
- As strings traduzidas não podem ser personalizadas; nós esperamos introduzir esse recurso no `texi2any` no futuro.

Além do último, nós não pretendemos reimplementar essas diferenças. Portanto, o caminho a seguir para autores(as) é o de alterar manuais e processos de construção conforme necessário para usar os novos recursos e métodos do `texi2any`. Os(As) mantenedores(as) do `texi2html` (um dos quais é o autor principal do `texi2any`) não pretendem fazer mais lançamentos.

Aqui está a tabela mostrando opções do `texi2html` e variáveis de personalização correspondentes do `texi2any`.

`--toc-links`	`TOC_LINKS`
`--short-ext`	`SHORTEXTN`
`--prefix`	`PREFIX`
`--short-ref`	`SHORT_REF`
`--idx-sum`	`IDX_SUMMARY`
`--def-table`	`DEF_TABLE`
`--ignore-preamble-text`	`IGNORE_PREAMBLE_TEXT`
`--html-xref-prefix`	`EXTERNAL_DIR`
`--l2h`	`L2H`
`--l2h-l2h`	`L2H_L2H`
`--l2h-skip`	`L2H_SKIP`
`--l2h-tmp`	`L2H_TMP`
`--l2h-file`	`L2H_FILE`
`--l2h-clean`	`L2H_CLEAN`
`--use-nodes`	`USE_NODES`
`--monolithic`	`MONOLITHIC`
`--top-file`	`TOP_FILE`
`--toc-file`	`TOC_FILE`
`--frames`	`FRAMES`
`--menu`	`SHOW_MENU`
`--debug`	`DEBUG`
`--doctype`	`DOCTYPE`
`--frameset-doctype`	`FRAMESET_DOCTYPE`
`--test`	`TEST`

Finalmente, quaisquer usuários(as) do `texi2html` buscando informações mais detalhadas pode verificar o arquivo de rascunho `doc/texi2oldapi.texi` no repositório do fonte do Texinfo. Ele consiste principalmente de notas muito grosseiras, mas ainda pode ser útil para alguns(mas).

21 Criando e Instalando Arquivos Info

Este capítulo descreve como criar e instalar arquivos Info. Veja Seção 1.3 [Arquivos do Info], Página 5, para informações gerais acerca do formato do arquivo em si.

21.1 Criando um Arquivo do Info

`makeinfo` é um programa que converte um arquivo do Texinfo em um arquivo do Info, em arquivo HTML ou em texto simples. `texinfo-format-region` e `texinfo-format-buffer` são funções do GNU Emacs que convertem Texinfo para Info.

Para informações acerca de como instalar o arquivo do Info no sistema Info, veja Seção 21.2 [Instalando Um Arquivo do Info], Página 188.

21.1.1 Vantagens do `makeinfo`

O utilitário `makeinfo` cria um arquivo do Info a partir de um fonte do Texinfo, fornecendo melhores mensagens de erro que qualquer um dos comandos de formatação do Emacs. Nós o recomendamos. O programa `makeinfo` é independente do Emacs. Você pode executar o `makeinfo` de três maneiras: a partir de um shell do sistema operacional; a partir de um shell dentro do Emacs; ou digitando o comando *C-c C-m C-r* ou *C-c C-m C-b* no modo Texinfo no Emacs.

Os comandos `texinfo-format-region` e `texinfo-format-buffer` possivelmente sejam úteis se você não puder executar `makeinfo`.

21.1.2 Executando `makeinfo` dentro do Emacs

Você pode executar `makeinfo` no modo Texinfo do GNU Emacs usando os comandos `makeinfo-region` ou `makeinfo-buffer`. No modo Texinfo, os comandos são vinculados a *C-c C-m C-r* e *C-c C-m C-b* por padrão.

C-c C-m C-r
M-x makeinfo-region
: Formata a região atual para Info.

C-c C-m C-b
M-x makeinfo-buffer
: Formata o buffer atual para Info.

Quando você invoca `makeinfo-region` a saída vai para um buffer temporário. Quando você invoca `makeinfo-buffer` a saída vai para o conjunto de arquivos com `@setfilename` (veja Seção 3.2.3 [`@setfilename`], Página 16).

Os comandos `makeinfo-region` e `makeinfo-buffer` do Emacs executam o programa `makeinfo` em um buffer temporário de shell. Se `makeinfo` encontrar quaisquer erros, o Emacs exibe as mensagens de erro no buffer temporário.

Você consegue analisar as mensagens de erro digitando *C-x `* (`next-error`). Isso faz com que o Emacs vá e posicione o cursor na linha no fonte do Texinfo que `makeinfo` pensa que causou o erro. Veja Seção "Running `make` or Compilers Generally" em *The GNU Emacs Manual*, para mais informações acerca de usar o comando `next-error`.

Além disso, você pode matar o shell no qual o comando `makeinfo` está executando ou fazer com que o buffer do shell exiba a saída dele mais recente.

C-c C-m C-k
M-x makeinfo-kill-job
: Mata o atual trabalho do `makeinfo` executando (a partir de `makeinfo-region` ou `makeinfo-buffer`).

C-c C-m C-l
M-x makeinfo-recenter-output-buffer
: Reexibe o buffer do shell do `makeinfo` para exibir a saída dele mais recente.

(Observe que os comandos paralelos para encerrar e recentralizar um trabalho do TEX são *C-c C-t C-k* e *C-c C-t C-l*. Veja Seção 19.6 [Impressão no Modo Texinfo], Página 155).

Você pode especificar opções para `makeinfo` configurando a variável `makeinfo-options` com o comando *M-x customize* ou *M-x set-variable*, ou configurando a variável em teu arquivo de inicialização `.emacs`.

Por exemplo, você poderia escrever o seguinte no teu arquivo `.emacs`:

```
(setq makeinfo-options
     "--paragraph-indent=0 --no-split
      --fill-column=70 --verbose")
```

Para mais informações, veja-se [Opções do `makeinfo`], Página 163, bem como "Easy Customization Interface", "Examining and Setting Variables" e "Init File" em *O Manual do GNU Emacs*.

21.1.3 Os Comandos `texinfo-format...`

No GNU Emacs em modo Texinfo, você pode formatar parte ou todo um arquivo do Texinfo com o comando `texinfo-format-region`. Isso formata a região atual e exibe o texto formatado em um buffer temporário chamado '`*Info Region*`'.

Similarmente, você pode formatar um buffer com o comando `texinfo-format-buffer`. Esse comando cria um novo buffer e gera o arquivo do Info nele. Digitar *C-x C-s* salvará o arquivo do Info sob o nome especificado pela linha `@setfilename`, que precisa estar perto do início do arquivo do Texinfo.

C-c C-e C-r
`texinfo-format-region`
: Formata a região atual para Info.

C-c C-e C-b
`texinfo-format-buffer`
: Formata o buffer atual para Info.

Os comandos `texinfo-format-region` e `texinfo-format-buffer` te fornecem alguma verificação de erros, e outras funções podem te fornecer mais ajuda para encontrar erros de formatação. Esses procedimentos são descritos em um anexo; veja-se Apêndice F [Capturando Erros], Página 252. Entretanto, o programa `makeinfo` fornece melhor verificação de erros (veja Seção 21.1.2 [`makeinfo` no Emacs], Página 185).

Uma peculiaridade dos comandos `texinfo-format-buffer` e `texinfo-format-region` é a de que eles não recuam (nem preenchem) parágrafos que contenham comandos `@w` ou `@*`.

21.1.4 Formatação em Lote

Você pode formatar arquivos do Texinfo para o Info usando `batch-texinfo-format` e o modo lote do Emacs. Você pode executar o Emacs no modo lote a partir de qualquer shell, incluindo um shell dentro do Emacs. (Veja Seção "Initial Options" em *O Manual do GNU Emacs*).

Aqui está um comando de shell para formatar todos os arquivos que terminam em `.texinfo` no diretório atual:

```
emacs -batch -funcall batch-texinfo-format *.texinfo
```

O Emacs processa todos os arquivos listados na linha de comando, mesmo se um erro ocorrer ao tentar formatar alguns deles.

Execute `batch-texinfo-format` somente com Emacs em modo lote, como mostrado; não é interativo. Ele mata o Emacs em modo lote ao concluir.

`batch-texinfo-format` é conveniente se você carecer do `makeinfo` e quiser formatar vários arquivos do Texinfo de uma vez. Quando usa o modo Lote, você cria um novo processo do Emacs. Isso libera teu Emacs atual, de forma que você continue trabalhando nele. (Quando executa `texinfo-format-region` ou `texinfo-format-buffer`, você não consegue usar esse Emacs para mais nada até que o comando termine).

21.1.5 Arquivos de Etiqueta e Arquivos de Divisão

Se um arquivo do Texinfo tiver mais que 30.000 bytes, `texinfo-format-buffer` cria automaticamente uma tabela de etiquetas para o arquivo dele do Info; `makeinfo` sempre cria uma tabela de etiquetas. Com uma *tabela de etiquetas*, Info pode pular para novos nós mais rapidamente que de outra forma.

Além disso, se o arquivo do Texinfo contiver mais que 300.000 bytes, `texinfo-format-buffer` e `makeinfo` dividem o arquivo grande do Info em subarquivos *indiretos* mais curtos de cerca de 300.000 bytes cada. Arquivos grandes são divididos em arquivos menores, de forma que o Emacs não precise fazer um buffer grande para armazenar todo um arquivo grande do Info; em vez disso, o Emacs aloca apenas memória suficiente para o arquivo pequeno e dividido que seja necessária no momento. Dessa forma, o Emacs evita desperdiçar memória quando você executa o Info. (Antes da divisão ser implementada, os arquivos do Info eram sempre mantidos curtos e *arquivos de inclusão* foram projetados como uma maneira de criar um manual impresso grande a partir dos arquivos menores do Info. Veja Capítulo 18 [Arquivos de Inclusão], Página 146, para mais informações. Os arquivos de inclusão ainda são usados para documentos muito grandes, como *The Emacs Lisp Reference Manual*, no qual cada capítulo é um arquivo separado).

Quando um arquivo é dividido, o próprio Info faz uso de uma versão abreviada do arquivo original que contém apenas a tabela de etiquetas e referências para os arquivos que foram divididos. Os arquivos divididos são chamados de arquivos *indiretos*.

Os arquivos divididos tem nomes que são criados pospondo '`-1`', '`-2`', '`-3`' e assim por diante no nome do arquivo especificado pelo comando `@setfilename`. A versão abreviada do arquivo original continua a ter o nome especificado por `@setfilename`.

Em um estágio da escrita deste documento, por exemplo, o arquivo do Info foi salvo como o arquivo `test-texinfo` e esse arquivo se parecia com isto:

```
Arquivo do Info: test-texinfo,    -*-Text-*-
produzido por texinfo-format-buffer
a partir do arquivo: new-texinfo-manual.texinfo

^_
Indireto:
test-texinfo-1: 102
test-texinfo-2: 50422
test-texinfo-3: 101300
^_^L
Tabela de etiquetas:
(Indireto)
Nó: overview^?104
Nó: info file^?1271
Nó: printed manual^?4853
Nó: conventions^?6855
...
```

(Mas `test-texinfo` tinha muito mais nós que os mostrados aqui). Cada um dos arquivos indiretos divididos, `test-texinfo-1`, `test-texinfo-2` e `test-texinfo-3`, está listado nesse arquivo depois da linha que diz '`Indireto:`'. A tabela de etiquetas está listada depois da linha que diz '`Tabela de Etiquetas:`'.

Na lista de arquivos indiretos, o número seguinte ao nome do arquivo registra o número cumulativo de bytes nos arquivos indiretos precedentes, não contando a lista de arquivos em si, a tabela de etiquetas ou qualquer texto de permissões no primeiro arquivo. Na tabela de etiquetas, o número seguinte ao nome do nó registra o local do início do nó, em bytes, a partir do início da saída (não dividida).

Se você estiver usando `texinfo-format-buffer` para criar arquivos do Info, você pode querer executar o comando `Info-validate`. (O comando `makeinfo` faz um trabalho tão bom sozinho, que você não precisa do `Info-validate`). No entanto, você não pode executar o comando de verificação de nó *`M-x Info-validate`* sobre arquivos indiretos. Para informações acerca de como evitar que arquivos sejam divididos e como validar a estrutura dos nós, veja-se Seção F.6.1 [Usando `Info-validate`], Página 256.

21.2 Instalando Um Arquivo do Info

Arquivos do Info geralmente são mantidos no diretório `info`. Você pode ler arquivos do Info usando o programa autônomo Info ou o leitor Info integrado ao Emacs. (Veja *Info*, para uma introdução ao Info).

21.2.1 O Arquivo de Diretório `dir`

Para que o Info funcione, o diretório `info` precisa conter um arquivo que sirva como um diretório de nível superior para o sistema Info. Por convenção, esse arquivo é chamado `dir`. (Você pode encontrar o local desse arquivo dentro do Emacs digitando *`C-h i`* para entrar no Info e então digitando *`C-x C-f`* para ver o caminho para o diretório `info`).

O arquivo `dir` é ele próprio um arquivo do Info. Ele contém o menu de nível superior para todos os arquivos do Info no sistema. O menu se parece com isto:

```
* Menu:
* Info:    (info).     Sistema de navegação de documentação.
* Emacs:   (emacs).    O editor de texto extensível e autodocumentado.
* Texinfo: (texinfo).  Com um arquivo fonte, crie um manual impresso usando @TeX{} ou u
...
```

Cada uma dessas entradas de menu aponta para o nó 'Top' do arquivo do Info que é nomeado entre parênteses. (A entrada de menu não precisa especificar o nó 'Top', pois o Info vai para o nó 'Top' se nenhum nome de nó for mencionado. Veja Seção 4.9.6 [Nós em Outros Arquivos do Info], Página 38).

Assim, a entrada '`Info`' aponta para o nó 'Top' do arquivo `info` e a entrada '`Emacs`' aponta para o nó 'Top' do arquivo `emacs`.

Em cada um dos arquivos do Info, o ponteiro 'Acima' do nó 'Top' remete de volta ao arquivo `dir`. Por exemplo, a linha para o nó 'Top' do manual do Emacs se parece com isto no Info:

```
File: emacs  Node: Top, Acima: (DIR), Próximo: Distrib
```

Nesse caso, o nome do arquivo `dir` é escrito em letras maiúsculas—pode ser escrito em letras maiúsculas ou minúsculas. Isso não é verdade em geral; é um caso especial para `dir`.

21.2.2 Listando um Novo Arquivo Info

Para adicionar um novo arquivo do Info ao teu sistema, você precisa escrever uma entrada de menu para adicionar ao menu no arquivo `dir` no diretório `info`. Por exemplo, se estivesse adicionando documentação para o GDB, você escreveria a seguinte nova entrada:

```
* GDB: (gdb).           O depurador C em nível de fonte.
```

A primeira parte da entrada do menu é o nome da entrada do menu, seguido por dois pontos. A segunda parte é o nome do arquivo do Info, entre parênteses, seguido por um ponto. A terceira parte é a descrição.

O nome de um arquivo do Info frequentemente tem uma extensão `.info`. Assim, o arquivo do Info para GDB pode ser chamado de `gdb` ou `gdb.info`. Os programas leitores do Info automaticamente tentam o nome do arquivo com e sem `.info`[1]; de forma que é melhor evitar desordem e não escrever '`.info`' explicitamente na entrada do menu. Por exemplo, a entrada do menu do GDB deveria usar apenas '`gdb`' para o nome do arquivo, não '`gdb.info`'.

21.2.3 Arquivos do Info em Outros Diretórios

Se um arquivo do Info não estiver no diretório `info`, existem três maneiras de se especificar o local dele:

1. Escreva o nome do caminho no arquivo `dir` como a segunda parte do menu.
2. Especifique o nome do diretório do Info na variável de ambiente `INFOPATH` em teu arquivo de inicialização `.profile` ou `.cshrc`. (Somente você e outros(as) que configuraram essa variável de ambiente poderão encontrar arquivos do Info cujo local seja especificado dessa maneira).
3. Se você estiver usando o Emacs, liste o nome do arquivo em um segundo arquivo `dir`, no diretório dele; e então adicione o nome desse diretório à variável `Info-directory-list` no teu arquivo de inicialização pessoal ou do sítio.

   Essa variável diz ao Emacs onde procurar por arquivos `dir` (os arquivos precisam ser nomeados `dir`). O Emacs mescla os arquivos nomeados `dir` provenientes de cada um dos diretórios listados. (Na versão 18 do Emacs, você consegue configurar a variável `Info-directory` para o nome de somente um diretório).

Por exemplo, para acessar um arquivo de teste no diretório `/home/bob/info`, você poderia adicionar uma entrada como esta ao menu no arquivo padrão `dir`:

```
* Teste: (/home/bob/info/info-test).  Arquivo de teste do próprio Bob.
```

Nesse caso, o nome absoluto do arquivo do arquivo `info-test` está escrito como a segunda parte da entrada do menu.

Se você não quiser editar o arquivo `dir` do sistema, você pode dizer ao Info onde procurar configurando a variável de ambiente `INFOPATH` em teu arquivo de inicialização do shell. Isso funciona com o Emacs e com leitores Info autônomos.

Especificamente, se usar um shell compatível com Bourne, como `sh` ou `bash` para teu interpretador de comandos de shell, você configura a variável de ambiente `INFOPATH` no arquivo de inicialização `.profile`; mas, se usar `csh` ou `tcsh`, você configura a variável no arquivo de inicialização `.cshrc`. Em sistemas MS-DOS/MS-Windows, você precisa configurar `INFOPATH` em teu arquivo `autoexec.bat` ou no registro. Cada tipo de shell usa uma sintaxe diferente.

- Em um arquivo `.cshrc`, você poderia configurar a variável `INFOPATH` conforme segue:

  ```
  setenv INFOPATH .:~/info:/usr/local/emacs/info
  ```

- Em um arquivo `.profile`, você alcançaria o mesmo efeito escrevendo:

  ```
  INFOPATH=.:$HOME/info:/usr/local/emacs/info
  export INFOPATH
  ```

- Em um arquivo `autoexec.bat`, você escreve este comando (observe o uso de '`;`' como separador de diretório e uma sintaxe diferente para usar valores de outras variáveis de ambiente):

  ```
  set INFOPATH=.;%HOME%/info;c:/usr/local/emacs/info
  ```

O '`.`' indica o diretório atual como de costume. O Emacs usa a variável de ambiente `INFOPATH` para inicializar o valor da variável `Info-directory-list` do próprio Emacs. O leitor Info autônomo mescla quaisquer arquivos chamados `dir` em qualquer diretório listado na variável `INFOPATH` em um menu unitário apresentado a você no nó chamado '`(dir)Top`'.

[1] Em sistemas MS-DOS/MS-Windows, o Info tentará a extensão `.inf` também.

No entanto, você configura `INFOPATH`, se o último caractere dela for dois pontos (em sistemas MS-DOS/MS-Windows, use um ponto e vírgula em vez disso), ele será substituído pelo caminho padrão (compilado). Isso te dá uma maneira de aumentar o caminho padrão com novos diretórios sem ter que listar todos os lugares padrão. Por exemplo (usando a sintaxe do `sh`):

```
INFOPATH=/home/bob/info:
export INFOPATH
```

pesquisará `/home/bob/info` primeiro, depois os diretórios padrão. Dois pontos iniciais ou duplos não são tratados especialmente.

Ao criar teu próprio arquivo `dir` para uso com `Info-directory-list` ou `INFOPATH`, é mais fácil começar copiando um arquivo `dir` existente e substituir todo o texto depois de '`* Menu:`' pelas tuas entradas desejadas. Dessa maneira, a pontuação e os caracteres especiais *CTRL*-_ que o Info precisa estarão presentes.

Como uma alternativa final, que funciona somente com o Info do Emacs, você pode mudar a variável `Info-directory-list`. Por exemplo:

```
(add-hook 'Info-mode-hook '(lambda ()
     (add-to-list 'Info-directory-list
  (expand-file-name "~/info"))))
```

21.2.4 Instalando Arquivos do Diretório do Info

Quando instalar um arquivo do Info no teu sistema, você pode usar o programa `install-info` para atualizar o arquivo `dir` de diretórios do Info. Normalmente o Makefile para o pacote executa `install-info`, logo depois de copiar o arquivo do Info para o local de instalação apropriado dele.

Para a finalidade de que o arquivo do Info funcione com `install-info`, você inclui os comandos `@dircategory` e `@direntry`...`@end direntry` no arquivo fonte do Texinfo. Use `@direntry` para especificar as entradas de menu a adicionar ao arquivo de diretórios do Info e use `@dircategory` para especificar em qual parte do diretório do Info colocá-lo. Aqui está como esses comandos são usados neste manual:

```
@dircategory Sistema de documentação Texinfo
@direntry
* Texinfo: (texinfo).           O formato da documentação GNU.
* install-info: (texinfo)Invocando install-info. ...
...
@end direntry
```

Aqui está o que isso produz no arquivo do Info:

```
INFO-DIR-SECTION Sistema de documentação Texinfo
START-INFO-DIR-ENTRY
* Texinfo: (texinfo).           O formato da documentação GNU.
* install-info: (texinfo)Invocando install-info. ...
...
END-INFO-DIR-ENTRY
```

O programa `install-info` vê essas linhas no arquivo do Info e assim é como ele sabe o que fazer.

Sempre use os comandos `@direntry` e `@dircategory` perto do começo da entrada do Texinfo, antes do primeiro comando `@node`. Se você usá-los mais tarde na entrada, `install-info` não os notará.

`install-info` automaticamente reformatará a descrição das entradas de menu que estiver adicionando. Como uma questão de convenção, a descrição da entrada principal (acima, '`O formato da documentação GNU`') deveria começar na coluna 32, começando em zero (como em `what-cursor-position` no Emacs). Isso fará com que ela se alinhe com a maioria das outras. A descrição para utilitários individuais deve começar na coluna 48, onde possível. Para mais

informações acerca de formatação, vejam-se as opções '`--calign`', '`--align`' e '`--max-width`' em Seção 21.2.5 [Invocando `install-info`], Página 191.

Se você usar `@dircategory` mais que uma vez no fonte do Texinfo, cada uso especificará a categoria 'atual'; quaisquer comandos `@direntry` subsequentes adicionarão a essa categoria.

Ao escolher um nome de categoria para o comando `@dircategory`, nós recomendamos consultar o Diretório de Software Livre (`http://www.gnu.org/directory`). Se teu programa não estiver listado lá, ou listado incorreta ou incompletamente, por favor, informe a situação para os(as) mantenedores(as) do diretório (`http://directory.fsf.org`), de forma que os nomes das categorias possam ser mantidos em sincronia.

Aqui estão alguns exemplos (veja-se o arquivo `util/dir-example` na distribuição do Texinfo para um grande arquivo `dir` de amostra):

Emacs
Localização
Impressão
Desenvolvimento de software
Bibliotecas de software
Criação e manipulação de texto

Cada nó 'Invoking' para cada programa instalado deveria ter um `@direntry` correspondente. Isso permite que usuários(as) encontrem facilmente a documentação para os diferentes programas que podem executar, como no tradicional sistema `man`.

21.2.5 Invocando `install-info`

`install-info` insere entradas de menu a partir de um arquivo do Info no arquivo `dir` de nível superior no sistema Info (vejam-se as seções anteriores para uma explicação de como o arquivo `dir` funciona). `install-info` também remove entradas de menu do arquivo `dir`. Ele é mais frequentemente executado como parte de instalação de software, ou ao construir um arquivo `dir` para todos os manuais em um sistema. Sinopse:

`install-info [`*`opção...`*`] [`*`arquivo-info`*` [`*`arquivo-dir`*`]]`

Se *arquivo-info*, ou *arquivo-dir* não for especificado, as opções (descritas abaixo) que os definem precisam ser. Não existem padrões de tempo de compilação, e a entrada padrão nunca é usada. `install-info` pode ler somente um arquivo do Info e escrever somente um arquivo `dir` por invocação.

Se *arquivo-dir* (como especificado) não existir, `install-info` o criará, se possível (sem entradas).

Se qualquer arquivo de entrada for comprimido com `gzip` (veja *Gzip*), `install-info` o descomprimirá automaticamente para leitura. E se *arquivo-dir* for comprimido, `install-info` também o deixará comprimido automaticamente depois de escrever quaisquer mudanças. Se *arquivo-dir* em si não existir, `install-info` tentará abrir *`arquivo-dir`*`.gz`, *`arquivo-dir`*`.xz`, *`arquivo-dir`*`.bz2`, *`arquivo-dir`*`.lz` e *`arquivo-dir`*`.lzma`, nessa ordem.

Opções:

`--add-once`
: Especifica que a entrada ou entradas serão colocadas somente em uma seção.

`--align=`*`coluna`*
: Especifica a coluna em que a segunda e as linhas subsequentes da descrição da entrada do menu serão formatadas para começar. O padrão para essa opção é '`35`'. Ela é usada em conjunto com a opção '`--max-width`'. *coluna* começa contagem em 1.

`--append-new-sections`
: Em vez de ordenar alfabeticamente novas seções, coloque-as ao final do arquivo DIR.

`--calign=`*coluna*
: Especifica a coluna em que a primeira linha da descrição da entrada do menu será formatada para começar. O padrão para essa opção é '`33`'. É usada em conjunto com a opção '`--max-width`'. Quando o nome da entrada do menu excede essa coluna, a descrição da entrada começará na linha seguinte. *coluna* começa contagem em 1.

`--debug`
: Informa o que está sendo feito.

`--delete`
: Deleta as entradas em *arquivo-info* de *arquivo-dir*. O nome do arquivo na entrada em *arquivo-dir* precisa ser *arquivo-info* (exceto por um '`.info`' opcional em qualquer um deles). Não insira quaisquer novas entradas. Quaisquer seções vazias que resultem da remoção também serão removidas.

`--description=`*texto*
: Especifica a parte explicativa da entrada do menu. Se você não especificar uma descrição (por meio de '`--entry`', '`--item`' ou dessa opção), a descrição será retirada do próprio arquivo do Info.

`--dir-file=`*nome*
: Especifica o nome do arquivo do arquivo de diretórios do Info. Isso é equivalente a usar o argumento *arquivo-dir*.

`--dry-run`
: Mesmo que '`--test`'.

`--entry=`*texto*
: Insere *texto* como uma entrada de diretórios do Info; *texto* deveria ter o formato de uma linha de item de menu do Info mais zero ou mais linhas extras começando com espaço em branco. Se você especificar mais que uma entrada, todas elas serão adicionadas. Se você não especificar quaisquer entradas, elas serão determinadas a partir de informações no próprio arquivo do Info.

`--help`
: Exibe uma mensagem de uso com o uso básico e todas as opções disponíveis e então sai com sucesso.

`--info-file=`*arquivo*
: Especifica o arquivo do Info para instalar no diretório. Isso é equivalente a usar o argumento *arquivo-info*.

`--info-dir=`*diretório*
: Especifica o diretório onde o arquivo de diretórios `dir` reside. Equivalente a '`--dir-file=`*diretório*`/dir`'.

`--infodir=`*diretório*
: Mesmo que '`--info-dir`'.

`--item=`*texto*
: Mesmo que '`--entry=`*texto*'. Uma entrada de diretórios do Info é, na verdade, um item de menu.

`--keep-old`
: Não substitui entradas de menu preexistentes. Quando '`--remove`' é especificado, essa opção significa que seções vazias não são removidas.

`--max-width=`*coluna*
: Especifica a coluna na qual a descrição da entrada do menu será quebrada. *coluna* começa contagem em 1.

`--maxwidth=`*coluna*
: Mesmo que '`--max-width`'.

`--menuentry=`*texto*
: Mesmo que '`--name`'.

`--name=`*texto*
: Especifica a parte do nome da entrada do menu. Se o *texto* não começar com um asterisco '`*`', é presumido ser o texto depois do '`*`' e antes dos parênteses que especificam o arquivo do Info. Caso contrário, *texto* é tomado literalmente e é tomado como definindo o texto até e incluindo o primeiro ponto (um espaço é posposto se necessário). Se você não especificar o nome (por meio de '`--entry`', '`--item`' ou dessa opção), ele será obtido do próprio arquivo do Info. Se o Info não contiver o nome, o nome base do arquivo do Info será usado.

`--no-indent`
: Suprime a formatação de novas entradas no arquivo `dir`.

`--quiet`
`--silent`
: Suprime avisos, etc., para operação silenciosa.

`--remove`
: Mesmo que '`--delete`'.

`--remove-exactly`
: Também como '`--delete`', mas somente entradas se o nome do arquivo do Info corresponder exatamente; os sufixos `.info` e (ou) `.gz` *não* são ignorados.

`--section=`*seção*
: Coloca as entradas desse arquivo na seção *seção* do diretório. Se você especificar mais que uma seção, todas as entradas serão adicionadas em cada uma das seções. Se você não especificar quaisquer seções, elas serão determinadas a partir de informações no próprio arquivo do Info. Se o arquivo do Info não especificar uma seção, as entradas do menu serão colocadas na seção Miscelâneas.

`--section` *expressãoregular seção*
: Mesmo que '`--regex=`*expressãoregular* `--section=`*seção* `--add-once`'.

  `install-info` tenta detectar quando essa sintaxe alternativa é usada, mas nem sempre adivinha corretamente. Aqui está a heurística que `install-info` usa:

  1. Se o segundo argumento para `--section` começar com um hífen, a sintaxe original será presumida.
  2. Se o segundo argumento para `--section` for um arquivo que possa ser aberto, a sintaxe original será presumida.
  3. Caso contrário, a sintaxe alternativa será usada.

  Quando a heurística falha porque o título da tua seção começa com um hífen, ou acontece de ser um nome de arquivo que pode ser aberto, a sintaxe deveria ser mudada para '`--regex=`*expressãoregular* `--section=`*seção* `--add-once`'.

`--regex=`*expressãoregular*
: Coloca as entradas desse arquivo em qualquer seção que corresponda a *expressãoregular*. Se mais que uma seção corresponder, todas as entradas serão adicionadas em cada uma das seções. Especifique *expressãoregular* usando a sintaxe básica de expressão regular, mais ou menos como usada com `grep`, por exemplo.

`--test`
: Suprime atualização do arquivo de diretórios.

`--version`
: Exibe informações da versão e sai com sucesso.

22 Gerando HTML

`makeinfo` gera saída do Info por padrão, mas dada a opção `--html`, ele gerará HTML, para navegadores da web e outros programas. Este capítulo fornece alguns detalhes acerca tal saída HTML.

`makeinfo` tem muitas variáveis de personalização definidas por usuário(a) com as quais você pode influenciar a saída HTML. Veja Seção 20.5 [Variáveis de Personalização], Página 170.

`makeinfo` também pode produzir saída nos formatos XML e Docbook, mas nós ainda não os descrevemos em detalhes. Veja Seção 1.2 [Formatos de Saída], Página 4, para uma breve visão geral de todos os formatos de saída.

22.1 Tradução de HTML

Primeiro, o HTML gerado por `makeinfo` é o HTML 4 padrão. Ele também tenta ser compatível com padrões anteriores (por exemplo, HTML 2.0, RFC-1866). Portanto, por favor, informe a saída proveniente de uma execução sem erros de `makeinfo` que tenha problemas práticos de portabilidade do navegador como um defeito (veja Seção 1.1 [Informando Defeitos], Página 3).

Algumas exceções conhecidas ao HTML 3.2 (usar '`--init-file=html32.pm`' produz uma saída de estrito HTML 3.2; veja Seção 20.2 [Invocando `texi2any`], Página 163):

1. As tabelas HTML 3.2 são geradas para o comando `@multitable` (veja Seção 9.5 [Tabelas Multi Colunas], Página 80), mas elas deveriam degradar razoavelmente em navegadores sem suporte a tabelas.
2. O atributo '`lang`' do HTML 4 no atributo '`<html>`' é usado.
3. Entidades que não estão no padrão HTML 3.2 também são usadas.
4. CSS é usado (veja Seção 22.3 [CSS de HTML], Página 195).
5. Alguns elementos do HTML 4 são usados: `thead`, `abbr`, `acronym`.

Para alcançar a máxima portabilidade e acessibilidade entre navegadores (tanto gráficos quanto baseados em texto), sistemas e usuários(as), a saída HTML é intencionalmente bem simples e genérica. Isso sempre tem sido nosso objetivo para usuários(as) estarem aptos(as) a personalizarem a saída aos desejos deles(as) via CSS (veja Seção 22.3 [CSS de HTML], Página 195) ou outros meios (veja Seção 20.5 [Variáveis de Personalização], Página 170. Se você não conseguir realizar uma personalização razoável, sinta-se à vontade para informar isso.

No entanto, nós não desejamos nos afastarmos do nosso objetivo básico de legibilidade mais ampla para a saída principal. Por exemplo, usar CSS sofisticado pode tornar possível que a saída HTML se assemelhe mais à saída do TEX em alguns detalhes, mas esse resultado não chega nem perto de compensar as dificuldades decorrentes.

Também não é intencionalmente nosso objetivo, e nem mesmo possível, passar por cada teste concebível de validação sem quaisquer diagnósticos. Testes diferentes de validação tem objetivos diferentes, frequentemente acerca da aplicação pedante de algum padrão ou outro. Nosso objetivo primordial é o de ajudar usuários(as), não cumprir cegamente os padrões.

Para repetir o que foi dito acima: por favor, informe a saída oriunda de uma execução sem erros do `makeinfo` que tenha problemas *práticos* de portabilidade do navegador como um defeito (veja Seção 1.1 [Informando Defeitos], Página 3).

Uns poucos outros pontos gerais acerca da saída HTML seguem.

Barra de navegação: Por padrão, uma barra de navegação é inserida no início de cada nó, análogo à saída do Info. Se a opção '`--no-headers`' for usada, a barra de navegação será inserida somente no início dos arquivos divididos. Elementos `<link>` do cabeçalho na saída dividida podem suportar navegação semelhante à Info com navegadores como Lynx e Emacs W3 que implementam esse recurso HTML 1.0.

Notas de rodapé: para HTML, quando o estilo de nota de rodapé for '`end`', ou se a saída não for dividida, as notas de rodapé são colocadas no final da saída. Se o estilo de nota de rodapé for configurado para '`separate`', e a saída for dividida, elas serão colocadas em um arquivo separado. Veja Seção 10.3.2 [Estilos de Notas de Rodapé], Página 87.

HTML bruto: `makeinfo` incluirá segmentos do fonte do Texinfo entre `@ifhtml` e `@end ifhtml` na saída HTML (mas não nenhum dos outros Condicionais, por padrão). O fonte entre `@html` e `@end html` é passado sem mudanças para a saída (ou seja, suprimindo a escapagem normal dos caracteres de entrada '`<`', '`>`' e '`&`', os quais tem significado especial em HTML). Veja Seção 16.1 [Comandos Condicionais], Página 128.

22.2 Divisão de HTML

Ao dividir a saída nos nós (que é o padrão), `makeinfo` escreve a saída HTML em (basicamente) um arquivo de saída por `@node` do fonte do Texinfo.

Cada nome de arquivo de saída é o nome do nó com espaços substituídos por '`-`' e caracteres especiais mudados para '`_`' seguidos pelo ponto de código deles em hexadecimal (veja Seção 22.4 [Xref de HTML], Página 197). Isso é para torná-lo portável e fácil de usar como um nome de arquivo. No caso incomum de dois nós terem o mesmo nome depois desse tratamento, eles serão escritos consecutivamente no mesmo arquivo, com âncoras HTML, de forma que cada um possa ser referenciado independentemente.

Se `makeinfo` for executado em um sistema que não distingue maiúsculas de minúsculas em nomes de arquivo, os nós que forem os mesmos, exceto por maiúsculas e minúsculas (por exemplo, '`index`' e '`Index`'), também serão guardados no mesmo arquivo de saída com âncoras. Você também pode fingir estar em um sistema de arquivos que não diferencia maiúsculas de minúsculas, configurando a variável de personalização `CASE_INSENSITIVE_FILENAMES`.

Também é possível dividir em capítulos ou seções com `--split` (veja Seção 20.2 [Invocando `texi2any`], Página 163). Nesse caso, os nomes dos arquivos são construídos depois do nome do nó associado ao relevante comando de seccionamento. Além disso, a menos que `--no-node-files` seja especificada, um arquivo de redirecionamento é gerado para cada nó para a finalidade de suportar mais confiavelmente referências cruzadas para esse manual (veja Seção 22.4 [Xref de HTML], Página 197).

Ao dividir, os arquivos de saída HTML são gravados em um subdiretório, com o nome escolhido conforme segue:

1. `makeinfo` primeiro tenta o subdiretório com o nome base oriundo de `@setfilename` (ou seja, qualquer extensão é removida). Por exemplo, a saída HTML para `@setfilename gcc.info` seria escrita em um subdiretório chamado '`gcc/`'.
2. Se esse diretório não puder ser criado por qualquer motivo, então `makeinfo` tentará pospor '`.html`' no nome do diretório. Por exemplo, a saída para `@setfilename texinfo` seria escrita em '`texinfo.html/`'.
3. Se o diretório '*nome*`.html`' também não puder ser criado, `makeinfo` desistirá.

Em qualquer caso, o arquivo de saída de nível superior dentro do diretório é sempre chamado '`index.html`'.

A saída monolítica (`--no-split`) é nomeada de acordo com `@setfilename` (com qualquer extensão '`.info`' substituída por '`.html`'), `--output` (o argumento é usado literalmente) ou baseada no nome do arquivo de entrada como último recurso (veja Seção 3.2.3 [`@setfilename`], Página 16).

22.3 CSS de HTML

Cascading Style Sheets (CSS para abreviar) é um padrão da Internet para influenciar a exibição de documentos HTML: veja-se `http://www.w3.org/Style/CSS/`.

Por padrão, `makeinfo` inclui alguns comandos CSS simples para implementar melhor a aparência de alguns ambientes do Texinfo. Aqui estão dois deles, como um exemplo:

```
pre.display { font-family:inherit }
pre.smalldisplay { font-family:inherit; font-size:smaller }
```

Uma explicação completa de CSS está (muito) além deste manual; por favor, veja-se a referência acima. Em resumo, no entanto, o acima diz ao navegador da web para usar um tamanho de fonte 'menor' para o texto de `@smalldisplay` e para usar a mesma fonte do documento principal para `@smalldisplay` e `@display`. Por padrão, o comando '`<pre>`' do HTML usa uma fonte mono-espaçada.

Você pode influenciar o CSS na saída HTML com duas opções do `makeinfo`: `--css-include=`*`arquivo`* e `--css-ref=`*`url`*.

A opção `--css-ref=`*`url`* adiciona a cada arquivo de saída HTML uma etiqueta '`<link>`' referenciando um CSS no *url* fornecido. Isso permite usar folhas de estilo externas. Você pode achar o arquivo `texi2html/examples/texinfo-bright-colors.css` útil para visualizar os elementos CSS na saída do Texinfo.

A opção `--css-include=`*`arquivo`* inclui o conteúdo *arquivo* na saída HTML, como você pode esperar. Entretanto, os detalhes são um pouco complicados, conforme descrito a seguir, para fornecer flexibilidade máxima.

O arquivo CSS pode começar com as assim chamadas diretivas '`@import`', as quais lincam especificações externas de CSS para navegadores usarem ao interpretar o documento. Novamente, uma descrição completa está além do nosso escopo aqui, mas nós descreveremos sintaticamente como elas funcionam, de forma que possamos explicar como `makeinfo` as manuseia.

Podem existir mais que um '`@import`', mas eles tem que vir primeiro no arquivo, com somente espaços em branco e comentários intercalados, sem definições normais. (Exceção técnica: uma diretiva '`@charset`' pode preceder os '`@import`'. Isso não altera o comportamento do `makeinfo`; ele apenas copia o '`@charset`' se presente). Comentários em arquivos CSS são delimitados por '`/* ... */`', como em C. Uma diretiva '`@import`' precisa estar em uma destas duas formas:

```
@import url(http://exemplo.org/foo.css);
@import "http://exemplo.net/bar.css";
```

No que diz respeito a `makeinfo`, os caracteres cruciais são o '`@`' no início e o ponto e vírgula terminando a diretiva. Ao ler o arquivo CSS, ele simplesmente copia qualquer diretiva '`@`' para a saída, como segue:

- Se *arquivo* contiver somente declarações normais CSS, ele será incluído depois do CSS padrão do `makeinfo`, assim substituindo-o.
- Se *arquivo* começar com as especificações '`@import`' (veja-se abaixo), então os '`import`' serão incluídos primeiro (eles tem de vir primeiro, de acordo com o padrão), e então o CSS padrão do `makeinfo` será incluído. Se você precisar substituir os padrões do `makeinfo` a partir de um '`@import`', você pode fazer isso com a construção '`! important`' do CSS, como em:

  ```
  pre.pequenoexemplo { font-size: inherit ! important }
  ```

- Se *arquivo* contiver especificações de '`@import`' e internas de CSS, as de '`@import`' serão incluídas primeiro, depois os padrões de `makeinfo` e, por último, a interna de CSS proveniente de *arquivo*.
- Qualquer diretiva @ diferente de '`@import`' e '`@charset`' é tratada como uma declaração CSS, significando que `makeinfo` inclui o CSS padrão dele e, em seguida, o restante do arquivo.

Se o arquivo CSS estiver malformado ou errôneo, a saída do `makeinfo` será inespecificada. O `makeinfo` não tenta interpretar o significado do arquivo CSS de maneira alguma; ele apenas

procura pelos caracteres especiais '`@`' e '`;`' e copia cegamente o texto para a saída. Comentários no arquivo CSS podem ou não ser incluídos na saída.

Além das possibilidades oferecidas pelo CSS, `makeinfo` tem muitas variáveis de personalização definíveis por usuário(a) com as quais você consegue influenciar a saída HTML. Veja Seção 20.5 [Variáveis de Personalização], Página 170.

22.4 Referências Cruzadas de HTML

Referências cruzadas entre manuais do Texinfo em formato HTML se tornam, no final, um link `<a>` padrão do HTML, mas os detalhes são infelizmente complexos. Esta seção descreve o algoritmo usado em detalhes, de forma que o Texinfo consiga cooperar com outros programas, tais como `texi2html`, escrevendo arquivos HTML mutuamente compatíveis.

Este algoritmo pode ou não ser usado para links *dentro* da saída HTML para um arquivo do Texinfo. Dado que nenhum problema de compatibilidade surge em tais casos, nós não precisamos especificar isso.

Nós tentamos suportar referências a esses manuais "externos" em formatos monolíticos e divididos. Um manual *monolítico* (mono) está inteiramente contido em um arquivo, e um manual *dividido* tem um arquivo para cada nó. (Veja Seção 22.2 [Divisão de HTML], Página 195).

O algoritmo foi desenvolvido principalmente por Patrice Dumas em 2003–04.

22.4.1 Fundamentos do Link de Referência Cruzada do HTML

Para nossos propósitos, um link do HTML consiste de quatro componentes: um nome de dispositivo, uma parte de diretório, uma parte de arquivo e uma parte de alvo. Nós sempre assumimos o protocolo `http`. Por exemplo:

`http://`*`dispositivo/diretório/arquivo`*`.html#`*`alvo`*

As informações para construir um link vem do nome do nó e do nome do manual no comando de referência cruzada no fonte do Texinfo (veja Capítulo 6 [Referências Cruzadas], Página 45) e de *informações externas* (veja Seção 22.4.6 [Configuração do Xref do HTML], Página 201).

Nós agora consideramos cada parte separadamente.

O *dispositivo* é programado para ser o dispositivo local. Isso poderia ser ou a string literal '`localhost`' ou, de acordo com as regras para links HTML, o '`http://localhost/`' poderia ser omitido completamente.

As partes *diretório* e *arquivo* são mais complicadas e dependem da natureza relativa de divisão/mono do manual sendo processado e do manual ao qual a referência cruzada se refere. A ideia subjacente é a de que existe um diretório para manuais do Texinfo em HTML, e um dado *manual* ou está disponível como um arquivo monolítico *`manual`*`.html` ou como um subdiretório dividido *`manual`*`/*.html`. Aqui estão os casos:

- Se o manual atual estiver dividido, e o manual referente também estiver dividido, o diretório será '*`../referente/`*' e o arquivo será o nome do nó expandido (descrito posteriormente).
- Se o manual atual estiver dividido e o manual referente estiver monolítico, o diretório será '`../`' e o arquivo será *`referente`*`.html`.
- Se o manual atual estiver monolítico e o manual referente estiver dividido, o diretório será *`referente/`* e o arquivo será o nome do nó expandido.
- Se o manual atual estiver monolítico e o manual referente também estiver monolítico, o diretório será `./` (ou apenas a string vazia), e o arquivo será *`referente`*`.html`.

Outra regra, que só vale para nomes de arquivo, é que nomes de arquivo base são truncados para 245 caracteres, para permitir para uma extensão ser posposta e ainda cumprir com o limite de 255 caracteres, o qual é comum a muitos sistemas de arquivo. Embora, tecnicamente, isso possa ser mudado com a variável de personalização `BASEFILENAME_LENGTH` (veja Seção 20.5.4

[Outras Variáveis de Personalização], Página 177), fazer isso tornaria referências entre manuais para tais nós inválidas.

Qualquer parte de diretório no argumento do nome de arquivo do comando fonte da referência cruzada é ignorada. Assim, `@xref{,,,../foo}` e `@xref{,,,foo}` ambos usam '`foo`' como o nome do manual. Isso é porque qualquer tal conexão rígida tentada do diretório dificilmente seja útil para ambas as saídas Info e HTML.

Finalmente, a parte *alvo* sempre é o nome do nó expandido.

Se o manual atual será dividido ou monolítico é determinado pela opção de usuário(a); `makeinfo` padroniza para dividir, com a opção `--no-split` substituindo isso.

Se o manual referente é dividido ou monolítico, no entanto, é outra parte da informação externa (veja Seção 22.4.6 [Configuração do Xref do HTML], Página 201). Por padrão, `makeinfo` usa o mesmo formato do manual referente como o manual atual.

Portanto, pode existir uma incompatibilidade entre o formato do manual referente que o software gerador assume e o formato em que ele realmente está presente. Veja Seção 22.4.5 [Incompatibilidade de Xref do HTML], Página 201.

22.4.2 Expansão de Nome de Nó de Referência Cruzada do HTML

Conforme mencionado na seção anterior, a parte chave do algoritmo de referência cruzada do HTML é a conversão de nomes de nós no fonte do Texinfo em strings adequadas para identificadores e nomes de arquivo do XHTML. As restrições são semelhantes para cada um: letras ASCII simples, números e os caracteres '`-`' e '`_`' são tudo o que pode ser usado. (Embora âncoras HTML possam conter a maioria dos caracteres, XHTML é mais restritivo).

Referências cruzadas no Texinfo podem se referir ou a nós ou a âncoras (veja Seção 6.8 [`@anchor`], Página 51). No entanto, âncoras são tratadas identicamente a nós nesse contexto, de forma que nós continuaremos a dizer nomes de "nós" por simplicidade.

Uma exceção especial: o nó Top (veja Seção 3.6 [O Nó Top], Página 23) sempre é mapeado para o arquivo `index.html`, para corresponder ao software do servidor web. Não obstante, o *alvo* do HTML é '`Top`'. Assim (no caso dividido):

```
@xref{Top,,, emacs, O Manual do GNU Emacs}.
⇒ <a href="emacs/index.html#Top">
```

1. As letras ASCII padrão (a-z e A-Z) não são modificadas. Todos os outros caracteres podem ser mudados conforme especificado abaixo.
2. Os números ASCII padrão (0-9) não são modificados, exceto quando um número é o primeiro caractere do nome do nó. Nesse caso, veja-se abaixo.
3. Vários espaços consecutivos, tabulações e caracteres de nova linha são transformados em apenas um espaço. (Não é possível ter novas linhas em nomes de nós com a implementação atual, mas nós especificamos isso de qualquer maneira, só por precaução).
4. Espaços iniciais e finais são removidos.
5. Depois que o acima tenha sido aplicado, cada caractere de espaço restante é convertido em um caractere '`-`'.
6. Outros caracteres ASCII de 7 bits são transformados em '`_00xx`', onde *xx* é o código de caractere ASCII em hexadecimal (minúsculo). Isso inclui '`_`', que é mapeado para '`_005f`'.
7. Se o nome do nó não começar com uma letra, a string literal '`g_t`' será prefixada ao resultado. (Devido às regras acima, essa string nunca pode ocorrer de outra forma; é uma escolha arbitrária, significando "GNU Texinfo"). Isso é necessário porque o XHTML exige que os identificadores comecem com uma letra.

Por exemplo:

```
@node Um  nó --- com _'%
```

⇒ `Um-nó-_002d_002d_002d-com-_005f_0027_0025`

Exemplo de traduções de caracteres comuns:

- '`_`' ⇒ '`_005f`'
- '`-`' ⇒ '`_002d`'
- '`Um nó`' ⇒ '`Um-nó`'

Em sistemas computacionais com conversão para minúsculas, nós que diferem somente pela caixa serão mapeados para o mesmo arquivo. Em particular, como mencionado acima, Top sempre mapeia para o arquivo `index.html`. Assim, em um sistema com conversão para minúsculas, Top e um nó chamado 'Index' serão ambos escritos em `index.html`. Felizmente, os alvos servem para distinguir esses casos, já que os nomes de alvos do HTML são sempre sensíveis à caixa, independente do sistema operacional.

22.4.3 Expansão do Comando de Referências Cruzadas do HTML

Os nomes dos nós podem conter comandos @ (veja Seção 4.4 [Exigências de Linha de Nó], Página 31). Esta seção descreve como eles são manuseados.

Primeiro, os comentários são removidos.

Em seguida, quaisquer comandos `@value` (veja Seção 16.5.1 [`@set @value`], Página 132) e invocações de macro (veja Seção 17.2 [Invocando Macros], Página 138) são totalmente expandidos.

Então, para os comandos a seguir, o nome e as chaves do comando é removido, e o texto do argumento é transformado recursivamente:

```
@asis @b @cite @code @command @dfn @dmn @dotless
@emph @env @file @i @indicateurl @kbd @key
@samp @sansserif @sc @slanted @strong @sub @sup
@t @U @var @verb @w
```

Para `@sc`, quaisquer letras são maiúsculas.

Além disso, os seguintes comandos são substituídos por texto constante, como mostrado abaixo. Se qualquer um desses comandos tiver argumentos não vazios, como em `@TeX{ruim}`, é um erro, e o resultado é inespecificado. Nesta tabela, '(espaço)' significa um caractere de espaço e '(nada)' significa a string vazia. A notação 'U+*hhhh*' significa ponto de código Unicode *hhhh* (em hexadecimal, como de costume).

Existem outras transformações de muitas dessas expansões para produzir o arquivo final ou outro nome de alvo, como caracteres de espaço para '`-`', etc., de acordo com as outras regras.

`@(newline)`	(espaço)
`@(space)`	(espaço)
`@(tab)`	(espaço)
`@!`	'`!`'
`@*`	(espaço)
`@-`	(nada)
`@.`	'`.`'
`@:`	(nada)
`@?`	'`?`'
`@@`	'`@`'
`@{`	'`{`'
`@}`	'`}`'
`@LaTeX`	'`LaTeX`'
`@TeX`	'`TeX`'
`@arrow`	U+2192
`@bullet`	U+2022

`@comma`	‘,’
`@copyright`	U+00A9
`@dots`	U+2026
`@enddots`	‘...’
`@equiv`	U+2261
`@error`	‘error-->’
`@euro`	U+20AC
`@exclamdown`	U+00A1
`@expansion`	U+21A6
`@geq`	U+2265
`@leq`	U+2264
`@minus`	U+2212
`@ordf`	U+00AA
`@ordm`	U+00BA
`@point`	U+2605
`@pounds`	U+00A3
`@print`	U+22A3
`@questiondown`	U+00BF
`@registeredsymbol`	U+00AE
`@result`	U+21D2
`@textdegree`	U+00B0
`@tie`	(espaço)

Comandos @ de aspas (`@quotedblright{}` e similares) são igualmente substituídos pelos valores Unicode deles. *Caracteres* de aspas normais (por exemplo, ‘ e ’ do ASCII) não são alterados. Veja Seção 12.5 [Inserindo Aspas], Página 101.

Quaisquer comandos `@acronym`, `@abbr`, `@email` e `@image` são substituídos pelo primeiro argumento deles. (Para esses comandos, todos os argumentos subsequentes são opcionais e ignorados aqui). Veja Seção 7.1.14 [`@acronym`], Página 62, e Seção 7.1.16 [`@email`], Página 63, e Seção 10.2 [Imagens], Página 84.

Acentos são manuseados de acordo com a próxima seção.

Qualquer outro comando é um erro e o resultado é inespecificado.

22.4.4 Expansão de Caracteres de 8 Bits das Referências Cruzadas do HTML

Usualmente, caracteres diferentes de ASCII simples de 7 bits são transformados no(s) correspondente(s) ponto(s) de código Unicode na Forma de Normalização C, que usa caracteres pré-compostos quando disponíveis. (Esse é a forma de normalização recomendada pelo W3C e outros órgãos). Isso vale quando esse ponto de código é `0xffff` ou menos, como quase sempre é.

Eles serão então transformados pelas regras acima na string ‘`_hhhh`’, onde *hhhh* é o ponto de código em hexadecimal.

Por exemplo, combinando esta regra e a seção anterior:

```
@node @b{A} @TeX{} @u{B} @point{}@enddots{}
⇒ A-TeX-B_0306-_2605_002e_002e_002e
```

Aviso: 1) `@enddots` expande para três pontos, os quais, por sua vez, expandem para três ‘`_002e`’; 2) `@u{B}` é um ‘B’ com um acento breve, que não existe como um caractere Unicode pré-acentuado, portanto expande para ‘`B_0306`’ (B com acento breve combinado).

Quando o ponto de código Unicode está acima de `0xffff`, a transformação é ‘`__xxxxxx`’, ou seja, dois sublinhados iniciais seguidos por seis dígitos hexadecimais. Como o Unicode declarou que o ponto de código mais alto deles é `0x10ffff`, isso é suficiente. (Nós sentimos que era melhor

definir esse escape extra que sempre usar seis dígitos hexadecimais, já que os dois primeiros seriam quase sempre zeros).

Esse método funciona bem se o nome do nó consistir principalmente de caracteres ASCII e contiver somente alguns de 8 bits. Mas, se o documento estiver escrito em um idioma cujo script não é baseado no alfabeto latino (por exemplo, ucraniano), ele criará nomes de arquivo consistindo quase inteiramente de notações '`_xxxx`', as quais são inconvenientes e quase ilegíveis. Para lidar com esses casos, `makeinfo` oferece a opção de linha de comando `--transliterate-file-names`. Essa opção habilita *transliteração* de nomes de nó em caracteres ASCII para os propósitos da criação e referenciamento de nome de arquivo. A transliteração é baseada em princípios fonéticos, o que torna os nomes gerados de arquivos mais facilmente compreensíveis.

Para a definição da Forma de Normalização C do Unicode, veja-se o informe Unicode UAX#15, `http://www.unicode.org/reports/tr15/`. Muitos documentos e implementações relacionados estão disponíveis em outros lugares na web.

22.4.5 Incompatibilidade de Referências Cruzadas do HTML

Conforme mencionado anteriormente (veja Seção 22.4.1 [Fundamentos do Link Xref do HTML], Página 197), o software gerador pode precisar adivinhar se um manual fornecido sendo referenciado cruzadamente está disponível em formato dividido ou monolítico—e, inevitavelmente, pode adivinhar errado. No entanto, quando o manual do *referente* é gerado, é possível lidar com pelo menos algumas incompatibilidades.

No caso onde nós assumimos que o referente é dividido, mas ele está realmente disponível em monolítico, o único recurso seria gerar um subdiretório `manual/` cheio de arquivos HTML que redirecionam de volta para o monolítico `manual.html`. Como isso é essencialmente o mesmo que um manual dividido em primeiro lugar, não é muito atraente.

Por outro lado, no caso onde nós assumimos que o referente é monolítico, mas ele está realmente disponível dividido, é possível usar JavaScript para redirecionar a partir do supostamente monolítico `manual.html` para os diferentes arquivos `manual/node.html`. Aqui está um exemplo:

```
function redirect() {
  switch (location.hash) {
    case "#Nodo1":
      location.replace("manual/Nodo1.html#Nodo1"); break;
    case "#Nodo2" :
      location.replace("manual/Nodo2.html#Nodo2"); break;
    ...
    default:;
  }
}
```

Então, na etiqueta `<body>` de `manual.html`:

```
<body onLoad="redirect();">
```

Mais uma vez, isso é algo que o software que gerou o manual *referente* tem que fazer com antecedência, não é algo que o software que gera a referência cruzada no manual atual pode controlar.

22.4.6 Configuração de Referência Cruzada do HTML: `htmlxref.cnf`

`makeinfo` lê um arquivo chamado `htmlxref.cnf` para reunir informações para referências cruzadas para outros manuais na saída HTML. Ele é procurado nos seguintes diretórios:

`./` (o diretório atual)

`./.texinfo/`
(sob o diretório atual)

`~/.texinfo/`
: (onde `~` é o diretório inicial do(a) usuário(a) atual)

sysconfdir`/texinfo/`
: (onde *sysconfdir* é o diretório de sistema da configuração especificado em tempo de compilação, por exemplo, `/usr/local/etc`)

datadir`/texinfo/`
: (da mesma forma especificado em tempo de compilação, por exemplo, `/usr/local/share`)

Todos os arquivos encontrados são usados, com entradas anteriores substituindo as posteriores. A distribuição do Texinfo inclui um arquivo padrão que lida com muitos manuais GNU; ele é instalado no último dos diretórios acima, ou seja, *datadir*`/texinfo/htmlxref.cnf`.

O arquivo é orientado a linhas. Linhas consistindo somente de espaços em branco são ignoradas. Comentários são indicados com um '`#`' no início de uma linha, opcionalmente precedidos por espaços em branco. Como '`#`' pode ocorrer em URLs (como quase qualquer caractere), ele não inicia um comentário.

Cada linha não vazia e sem comentário precisa ser ou uma *atribuição de variável* ou uma *informação de manual*.

Uma linha de atribuição de variável se parece com isto:

```
nomevariável = valorvariável
```

Espaço em branco ao redor de '`=`' é opcional e ignorado. *nomevariável* deveria consistir de letras; maiúsculas e minúsculas são significativas. O *valorvariável* é uma string arbitrária, continuando até o fim da linha. As variáveis são então referenciadas com '`${`*nomevariável*`}`'; referências de variáveis podem ocorrer no *valorvariável*.

Uma linha de informação de manual se parece com isto:

```
manual palavrachave prefixourl
```

com *manual* o identificador curto para um manual, *palavrachave* sendo um de: `mono`, `node`, `section`, `chapter` e *prefixourl* descritos abaixo. Referências de variáveis podem ocorrer somente no *prefixourl*. Por exemplo (usado no `htmlxref.cnf` canônico):

```
G = http://www.gnu.org
GS = ${G}/software
hello mono    ${GS}/hello/manual/hello.html
hello chapter ${GS}/hello/manual/html_chapter/
hello section ${GS}/hello/manual/html_section/
hello node    ${GS}/hello/manual/html_node/
```

Se a palavra-chave for `mono`, *prefixourl* fornecerá o dispositivo, o diretório e o nome do arquivo para *manual* como um arquivo monolítico.

Se a palavra-chave for `node`, `section` ou `chapter`, *prefixourl* fornecerá o dispositivo e o diretório para *manual* dividido em nós, seções ou capítulos, respectivamente.

Quando disponível, `makeinfo` usará o valor "corresponding" para referências cruzadas entre manuais. Isto é, ao gerar saída monolítica (`--no-split`), a URL `mono` será usada; ao gerar saída que seja dividida por nó, a URL `node` será usada, etc. No entanto, se um manual não estiver disponível nesse formato, qualquer coisa que esteja disponível pode ser usada. Aqui está a ordem de pesquisa para cada estilo:

```
node    ⇒ node,    section, chapter, mono
section ⇒ section, chapter, node,    mono
chapter ⇒ chapter, section, node,    mono
mono    ⇒ mono,    chapter, section, node
```

Essas referências entre manuais em nível de seção e de capítulo só podem ser bem-sucedidas quando o manual alvo foi criado usando `--node-files`; esse é o padrão para saída dividida.

Se você tiver adições ou correções para o `htmlxref.cnf` distribuído com o Texinfo, por favor, contate `bug-texinfo@gnu.org` como de costume. Você consegue obter a versão mais recente em `http://ftpmirror.gnu.org/texinfo/htmlxref.cnf`.

22.4.7 Preservação de Link de Referência Cruzada do HTML: *manual*-`noderename.cnf`

Ocasionalmente, mudanças em um programa exigem remover (ou renomear) nós no manual para a finalidade de ter a melhor documentação. Dada a natureza da web, no entanto, links podem existir em qualquer lugar para um nó removido (renomear parece o mesmo que remover para esse propósito), e não é ideal que esses links simplesmente quebrem.

Portanto, o Texinfo fornece uma maneira para autores(as) de manuais especificarem nomes antigos de nós e os novos nós para os quais os nomes antigos deveriam ser redirecionados, por meio do arquivo *manual*-`noderename.cnf`, onde *manual* é o nome base do manual. Por exemplo, o manual `texinfo.texi` seria complementado por um arquivo `texinfo-noderename`.cnf. (Esse nome pode ser substituído configurando-se a variável de personalização `RENAMED_NODES_FILE`; veja Seção 20.5 [Variáveis de Personalização], Página 170).

O arquivo é lido em pares de linhas, conforme segue:

```
nome-antigo-nó
@@{} nome-novo-nó
```

A conversão usual de nomes de nós do Texinfo para nomes do HTML é aplicada; veja-se esta seção inteira para detalhes (veja Seção 22.4 [Xref de HTML], Página 197). O separador incomum '`@@{}`' é usado porque ele não é uma construção válida do Texinfo, de forma que não pode aparecer nos nomes de nós.

O efeito é que `makeinfo` gera um redirecionamento de *nome-antigo-nó* para *nome-novo-nó* ao produzir saída HTML. Assim, links externos para o nó antigo são preservados.

Linhas consistindo somente de espaços em branco são ignoradas. Comentários são indicados com um '`@c`' no início de uma linha, opcionalmente precedidos por espaços em branco.

Outra abordagem para preservar links para nós deletados ou renomeados é a de usar âncoras (veja Seção 6.8 [`@anchor`], Página 51). Não existe diferença efetiva entre as duas abordagens.

Apêndice A Detalhes do Comando @

Aqui estão os detalhes dos comandos @: informações acerca da sintaxe deles, uma lista de comandos e informações relativas a onde os comandos podem aparecer.

A.1 Sintaxe do Comando @

O Texinfo tem os seguintes tipos de comando @:

1. Comandos de chaves
   Esses comandos começam com @ seguido por uma letra ou uma palavra, seguida por um argumento entre chaves. Por exemplo, o comando `@dfn` indica o uso introdutório ou definidor de um termo; ele é usado conforme segue: '`No Texinfo, comandos @@ são comandos de @dfn{marcação}`'.

2. Comandos de linha
   Esses comandos ocupam uma linha inteira. A linha começa com @, seguido pelo nome do comando (uma palavra); por exemplo, `@center` ou `@cindex`. Se nenhum argumento for necessário, a palavra é seguida pelo fim da linha. Se existir um argumento, ele é separado do nome do comando por um espaço. Chaves não são usadas.

3. Comandos de bloco
   Esses comandos são escritos no início de uma linha, com texto geral nas linhas seguintes, terminados por um comando `@end` correspondente em uma linha própria. Por exemplo, `@example`, depois as linhas de um exemplo de codificação, depois `@end example`. Alguns desses comandos de bloco recebem argumentos como os comandos de linha; por exemplo, `@enumerate A` abrindo um ambiente terminado por `@end enumerate`. Aqui '`A`' é o argumento.

4. Comandos de inserção de símbolos sem argumentos
   Esses comandos começam com @ seguido por uma palavra seguida por uma chave esquerda e direita. Esses comandos inserem símbolos especiais no documento; eles não recebem argumentos. Alguns exemplos: `@dots{}` ⇒ '...', `@equiv{}` ⇒ '≡', `@TeX{}` ⇒ 'TEX' e `@bullet{}` ⇒ '•'.

5. Comandos não alfabéticos
   Os nomes dos comandos em todas as categorias acima consistem de caracteres alfabéticos, quase inteiramente em letras minúsculas. Ao contrário daqueles, os comandos não alfabéticos consistem de um @ seguido por um sinal de pontuação ou outro caractere que não é parte do alfabeto latino. Os comandos não alfabéticos quase sempre são parte do texto dentro de um parágrafo. Os comandos não alfabéticos incluem `@@`, `@{`, `@}`, `@.`, `@`*`ESPAÇO`* e a maioria dos comandos de acento.

6. Comandos diversos
   Existe um punhado de comandos que não se encaixam em nenhuma das categorias acima; por exemplo, o comando obsoleto `@refill`, que sempre é usado no final de um parágrafo imediatamente seguinte ao ponto final ou outro caractere de pontuação. `@refill` não recebe argumentos e não exige chaves. Da mesma forma, `@tab` usado em um bloco `@multitable` não recebe argumentos e não é seguido por chaves.

Assim, os comandos alfabéticos se enquadram em classes que tem diferentes sintaxes de argumento. Você não pode dizer a qual classe um comando pertence pela aparência do nome dele, mas pode dizer pelo significado do comando: se o comando representa um glifo, ele está na classe 4 e não exige um argumento; se faz sentido usar o comando entre outros textos como parte de um parágrafo, o comando está na classe 1 e precisa ser seguido por um argumento

entre chaves. Os comandos não alfabéticos, como `@:`, são exceções à regra; eles não precisam de chaves.

O propósito de ter sintaxe diferente para comandos é o de tornar os arquivos do Texinfo mais fáceis de ler e também ajudar os comandos de parágrafo e preenchimento do GNU Emacs a funcionarem corretamente.

A.2 Lista de Comandos @

Aqui está uma lista alfabética dos comandos @ no Texinfo. Colchetes, `[ ]`, indicam argumentos opcionais; reticências, '`...`', indicam texto repetido.

`@`*espaço em branco*
: Um `@` seguido por um espaço, tabulação ou nova linha produz um espaço entre palavras normal e extensível. Veja Seção 12.3.1 [Espaços Múltiplos], Página 97.

`@!`
: Produz um ponto de exclamação que finaliza uma frase (geralmente depois de uma letra maiúscula de fim de frase). Veja Seção 12.3.3 [Finalizando Uma Frase], Página 98.

`@"`

`@'`
: Gera um trema ou acento agudo, respectivamente, sobre o próximo caractere, como em ö e ó. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@*`
: Força uma quebra de linha. Veja Seção 13.2 [Quebras de Linha], Página 110.

`@,{`*c*`}`
: Gera um acento de cedilha sob *c*, como em ç. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@-`
: Insira um ponto de hifenização discricionário. Veja Seção 13.3 [`@- @hyphenation`], Página 111.

`@.`
: Produz um ponto que finaliza uma frase (geralmente depois de uma letra maiúscula de fim de frase). Veja Seção 12.3.3 [Finalizando Uma Frase], Página 98.

`@/`
: Não produz saída, mas permite uma quebra de linha. Veja Seção 13.2 [Quebras de Linha], Página 110.

`@:`
: Diz ao TEX para evitar inserir espaços em branco extras depois de um ponto, ponto de interrogação, ponto de exclamação ou dois pontos imediatamente anterior, como o TEX normalmente faria. Veja Seção 12.3.2 [Não Finalizando Uma Frase], Página 98.

`@=`
: Gera um acento de mácron (barra) sobre o próximo caractere, como em ō. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@?`
: Produz um ponto de interrogação que finaliza uma frase (geralmente depois de uma letra maiúscula de fim de frase). Veja Seção 12.3.3 [Finalizando Uma Frase], Página 98.

`@@`

`@atchar{}`
: Insere um sinal de arroba, '`@`'. Veja Seção 12.1.1 [Inserindo um Símbolo Arroba], Página 95.

`@\`

`@backslashchar{}`
: Insere uma barra invertida, '`\`'; `@backslashchar{}` funciona em qualquer lugar, enquanto `@\` funciona somente dentro de `@math`. Veja Seção 12.1.4 [Inserindo uma Barra Invertida], Página 96, e Seção 12.7 [Inserindo Fórmulas Matemáticas], Página 102.

`@^`
`@`` Gera um acento circunflexo (chapéu) ou grave, respectivamente, sobre o próximo caractere, como em ô e è. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@{`
`@lbracechar{}`
Insere uma chave esquerda, '`{`'. Veja Seção 12.1.2 [Inserindo Chaves], Página 95.

`@}`
`@rbracechar{}`
Insere uma chave direita, '`}`'. Veja Seção 12.1.2 [Inserindo Chaves], Página 95.

`@~` Gera um acento til sobre o próximo caractere, como em Ñ. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@AA{}`
`@aa{}` Gera as letras maiúsculas e minúsculas do anel A escandinavo, respectivamente: Å, å. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@abbr{`*abbreviation*`}`
Indica uma abreviação geral, como 'Comput.'. Veja Seção 7.1.13 [`@abbr`], Página 62.

`@acronym{`*acronym*`}`
Indica uma sigla em letras todas maiúsculas, como 'NASA'. Veja Seção 7.1.14 [`@acronym`], Página 62.

`@AE{}`
`@ae{}` Gera as ligaduras AE maiúsculas e minúsculas, respectivamente: Æ, æ. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@afivepaper`
Muda as dimensões da página para o tamanho de papel A5. Veja Seção 19.12 [Papel A4], Página 159.

`@afourlatex`
`@afourpaper`
`@afourwide`
Muda as dimensões da página para o tamanho de papel A4. Veja Seção 19.12 [Papel A4], Página 159.

`@alias` *novo=existente*
Torna o comando '`@`*novo*' um sinônimo para o comando existente '`@`*existente*'. Veja Seção 17.4 [`@alias`], Página 142.

`@allowcodebreaks` *verdadeiro-falso*
Controla quebra em '`-`' e '`_`' no TEX. Veja Seção 13.4 [`@allowcodebreaks`], Página 111.

`@anchor{`*nome*`}`
Define *nome* como o local atual para uso como um alvo de referência cruzada. Veja Seção 6.8 [`@anchor`], Página 51.

`@appendix` *título*
Inicia um anexo. O título aparece no sumário. No Info, o título é sublinhado com asteriscos. Veja Seção 5.4 [`@unnumbered @appendix`], Página 40.

`@appendixsec` *título*
`@appendixsection` *título*
Inicia uma seção de anexos dentro de um anexo. O título da seção aparece no sumário. No Info, o título é sublinhado com sinais de igual. `@appendixsection` é

uma grafia mais longa do comando `@appendixsec`. Veja Seção 5.7 [`@unnumberedsec @appendixsec @heading`], Página 41.

`@appendixsubsec` *título*
: Inicia uma subseção de anexos. O título aparece no sumário. No Info, o título é sublinhado com hifens. Veja Seção 5.9 [`@unnumberedsubsec @appendixsubsec @subheading`], Página 42.

`@appendixsubsubsec` *título*
: Inicia uma subsubseção de anexos. O título aparece no sumário. No Info, o título é sublinhado com pontos. Veja Seção 5.10 [`@subsubsection`], Página 42.

`@arrow{}` Gera um glifo de seta para a direita: '→'. Usado por padrão para `@click`. Veja Seção 12.9.8 [Sequências de Clique], Página 108.

`@asis` Usado seguindo `@table`, `@ftable` e `@vtable` para imprimir a primeira coluna da tabela sem realçar ("como está"). Veja [`@asis`], Página 78.

`@author` *autor(a)*
: Tipografa *autor(a)* alinhado à esquerda e sublinhado. Veja Seção 3.4.3 [`@title @subtitle @author`], Página 20.

`@b{`*texto*`}`
: Configura *texto* em uma fonte **negrito**. Sem efeito no Info. Veja Seção 7.2.3 [Fontes], Página 65.

`@bullet{}`
: Gera um ponto redondo grande, • ('`*`' no Info). Geralmente usado com `@table`. Veja Seção 12.8.5 [`@bullet`], Página 104.

`@bye` Para de formatar um arquivo. Os formatadores não veem nada no arquivo de entrada seguinte a `@bye`. Veja Seção 3.8 [Finalizando um Arquivo], Página 28.

`@c` *comentário*
: Inicia um comentário no Texinfo. O restante da linha não aparece em nenhuma saída. Um sinônimo para `@comment`. *DEL* também inicia um comentário. Veja Seção 2.2 [Comentários], Página 10.

`@caption` Define a legenda completa para um `@float`. Veja Seção 10.1.2 [`@caption @shortcaption`], Página 83.

`@cartouche`
: Destaca um exemplo ou citação desenhando uma caixa com cantos arredondados ao redor. Emparelha com `@end cartouche`. Sem efeito no Info. Veja Seção 8.14 [`@cartouche`], Página 73.

`@center` *linha-de-texto*
: Centraliza a linha de texto seguindo o comando. Veja Seção 3.4.2 [`@titlefont @center @sp`], Página 19.

`@centerchap` *linha-de-texto*
: Como `@chapter`, mas centraliza o título do capítulo. Veja Seção 5.3 [`@chapter`], Página 40.

`@chapheading` *título*
: Imprime um título não numerado, como um capítulo, mas omite da tabela de conteúdo. No Info, o título é sublinhado com asteriscos. Veja Seção 5.5 [`@majorheading @chapheading`], Página 41.

`@chapter` *título*
: Inicia um capítulo numerado. O título do capítulo aparece na tabela de conteúdo. No Info, o título é sublinhado com asteriscos. Veja Seção 5.3 [`@chapter`], Página 40.

`@cindex` *entrada*
: Adiciona *entrada* ao índice de conceitos. Veja Seção 11.3 [Definindo as Entradas de um Índice], Página 90.

`@cite{`*referência*`}`
: Destaca o nome de um livro ou de outra referência que não tenha um arquivo complementar do Info. Veja Seção 6.11 [`@cite`], Página 55.

`@clear` *sinalizador*
: Desconfigura *sinalizador*, impedindo que os comandos de formatação do Texinfo formatem o texto entre pares subsequentes de comandos `@ifset` *sinalizador* e `@end ifset`, e impedindo que `@value{`*sinalizador*`}` expanda para o valor para o qual *sinalizador* está configurada. Veja Seção 16.5 [`@set @clear @value`], Página 132.

`@click{}`
: Representa um "clique" em uma GUI. Usado dentro de `@clicksequence`. Veja Seção 12.9.8 [Sequências de Clique], Página 108.

`@clicksequence{`*ação* `@click{}` *ação*`}`
: Representa uma sequência de cliques em uma GUI. Veja Seção 12.9.8 [Sequências de Clique], Página 108.

`@clickstyle @`*comando*
: Executa @*comando* para cada `@click`; o padrão é `@arrow`. As chaves vazias usuais seguintes a @*comando* são omitidas. Veja Seção 12.9.8 [Sequências de Clique], Página 108.

`@code{`*código-amostra*`}`
: Indica uma expressão, um token sintaticamente completo de um programa ou um nome de programa. Sem aspas na saída do Info. Veja Seção 7.1.2 [`@code`], Página 57.

`@codequotebacktick` *ligado-desligado*
`@codequoteundirected` *ligado-desligado*
: Controla a saída de `` ` `` e `'` em exemplos de código. Veja Seção 12.2 [Inserindo Caracteres de Citação], Página 97.

`@comma{}`
: Insere um caractere vírgula ','; necessário somente quando uma vírgula literal fosse recebida como separador de argumentos. Veja Seção 12.1.3 [Inserindo Uma Vírgula], Página 95.

`@command{`*nome-comando*`}`
: Indica um nome de comando, como `ls`. Veja Seção 7.1.10 [`@command`], Página 61.

`@comment` *comentário*
: Inicia um comentário no Texinfo. O restante da linha não aparece em nenhuma saída. Um sinônimo para `@c`. Veja Seção 2.2 [Comentários], Página 10.

`@contents`
: Imprime uma tabela completa de conteúdo. Não tem efeito no Info, que usa menus. Veja Seção 3.5 [Gerando Um Sumário], Página 22.

`@copying`
: Especifica os(as) titulares dos direitos autorais e as condições de cópia para o documento. Emparelha com `@end cartouche`. Veja Seção 3.3.1 [`@copying`], Página 17.

`@copyright{}`
: Gera o símbolo de direitos autorais ©. Veja Seção 12.8.2 [`@copyright`], Página 104.

`@defcodeindex` *nome-índice*

Define um novo índice e comando dele de indexação. Imprime entradas em uma fonte do `@code`. Veja Seção 11.6 [Definindo Novos Índices], Página 93.

`@defcv` *categoria classe nome*
`@defcvx` *categoria classe nome*

Formata uma descrição para uma variável associada com uma classe em programação orientada a objetos. Recebe três argumentos: a categoria da coisa sendo definida, a classe à qual ela pertence e o nome dela. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@deffn` *categoria nome argumentos...*
`@deffnx` *categoria nome argumentos...*

Formata uma descrição para uma função, comando interativo ou entidade similar que pode receber argumentos. `@deffn` recebe como argumentos a categoria da entidade sendo descrita, o nome dessa entidade em particular e os argumentos dela, se existirem. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defindex` *nome-índice*

Define um novo índice e o comando dele de indexação. Imprime entradas em uma fonte romana. Veja Seção 11.6 [Definindo Novos Índices], Página 93.

`@definfoenclose` *novocomando, antes, depois*

Precisa ser usado dentro de `@ifinfo`; cria um novo comando `@`*novocomando* para Info que marca o texto cercando-o entre strings que precedem e seguem o texto. Veja Seção 17.5 [`@definfoenclose`], Página 142.

`@defivar` *classe instância-variável-nome*
`@defivarx` *classe instância-variável-nome*

Formata uma descrição para uma variável de instância em programação orientada a objetos. O comando é equivalente a '`@defcv {Variável de Instância} ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defmac` *nomemacro argumentos...*
`@defmacx` *nomemacro argumentos...*

Formata uma descrição para uma macro; equivalente a '`@deffn Macro ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defmethod` *classe nome-método argumentos...*
`@defmethodx` *classe nome-método argumentos...*

Formata uma descrição para um método em programação orientada a objetos; equivalente a '`@defop Método ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defop` *categoria classe nome argumentos...*
`@defopx` *categoria classe nome argumentos...*

Formata uma descrição para uma operação em programação orientada a objetos. `@defop` recebe como argumentos o nome da categoria da operação, o nome da classe da operação, o nome da operação e os argumentos dela, se existirem. Veja Capítulo 14 [Comandos de Definição], Página 114, e Seção 14.5.6 [Objetos Abstratos], Página 121.

`@defopt` *nome-opção*
`@defoptx` *nome-opção*

Formata uma descrição para uma opção de usuário(a); equivalente a '`@defvr {Opção de Usuário(a)} ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defspec` *nome-forma-especial argumentos...*
`@defspecx` *nome-forma-especial argumentos...*

Formata uma descrição para uma forma especial; equivalente a '`@deffn {Forma Especial} ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@deftp` *categoria nome-do-tipo atributos...*
`@deftpx` *categoria nome-do-tipo atributos...*

Formata uma descrição para um tipo de dado; os argumentos dela são a categoria, o nome do tipo (por exemplo, '`int`') e, em seguida, os nomes dos atributos dos objetos desse tipo. Veja Capítulo 14 [Comandos de Definição], Página 114, e Seção 14.5.5 [Tipos de Dados], Página 120.

`@deftypecv` *categoria classe tipo-dado nome*
`@deftypecvx` *categoria classe tipo-dado nome*

Formata uma descrição para uma variável de classe tipada em programação orientada a objetos. Veja Capítulo 14 [Comandos de Definição], Página 114, e Seção 14.5.6 [Objetos Abstratos], Página 121.

`@deftypefn` *categoria tipo-dado nome argumentos...*
`@deftypefnx` *categoria tipo-dado nome argumentos...*

Formata uma descrição para uma função ou para entidade similar que pode receber argumentos e que é tipada. `@deftypefn` recebe como argumentos a categoria da entidade sendo descrita, o tipo, o nome da entidade e os argumentos dela, se existirem. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@deftypefnnewline` *ligado-desligado*

Especifica se os tipos de retorno para `@deftypefn` e similares são impressos sozinhos nas linhas; padrão é desligado. Veja Seção 14.5.3 [Funções em Linguagens Tipadas], Página 118.

`@deftypefun` *tipo-dado nome-função argumentos...*
`@deftypefunx` *tipo-dado nome-função argumentos...*

Formata uma descrição para uma função em uma linguagem tipada. O comando é equivalente a '`@deftypefn Função ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@deftypeivar` *classe tipo-dado nome-variável*
`@deftypeivarx` *classe tipo-dado nome-variável*

Formata uma descrição para uma variável de instância tipada em programação orientada a objetos. Veja Capítulo 14 [Comandos de Definição], Página 114, e Seção 14.5.6 [Objetos Abstratos], Página 121.

`@deftypemethod` *classe tipo-dado nome-método argumentos...*
`@deftypemethodx` *classe tipo-dado nome-método argumentos...*

Formata uma descrição para um método tipado em programação orientada a objetos. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@deftypeop` *categoria classe tipo-dado nome argumentos...*
`@deftypeopx` *categoria classe tipo-dado nome argumentos...*

Formata uma descrição para uma operação tipada em programação orientada a objetos. Veja Capítulo 14 [Comandos de Definição], Página 114, e Seção 14.5.6 [Objetos Abstratos], Página 121.

`@deftypevar` *tipo-dado nome-variável*
`@deftypevarx` *tipo-dado nome-variável*

Formata uma descrição para uma variável em uma linguagem tipada. O comando é equivalente a '`@deftypevr Variável ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@deftypevr` *categoria tipo-dado nome*
`@deftypevrx` *categoria tipo-dado nome*
: Formata uma descrição para algo como uma variável em uma linguagem tipada—uma entidade que registra um valor. Recebe como argumentos a categoria da entidade sendo descrita, o tipo e o nome da entidade. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defun` *nome-função argumentos...*
`@defunx` *nome-função argumentos...*
: Formata uma descrição para uma função; equivalente a '`@deffn Função ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defvar` *nome-variável*
`@defvarx` *nome-variável*
: Formata uma descrição para uma variável; equivalente a '`@defvr Variável ...`'. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@defvr` *categoria nome*
`@defvrx` *categoria nome*
: Formata uma descrição para qualquer tipo de variável. `@defvr` recebe como argumentos a categoria da entidade e o nome da entidade. Veja Capítulo 14 [Comandos de Definição], Página 114.

`@detailmenu`
: Marca a listagem detalhada (opcional) de nós em um menu mestre. Veja Seção 3.6.2 [Partes do Menu Mestre], Página 24.

`@dfn{`*termo*`}`
: Indica o uso introdutório ou definidor de um termo. Veja Seção 7.1.12 [`@dfn`], Página 62.

`@DH{}`
`@dh{}` Gera a letra maiúscula e minúscula islandesa eth, respectivamente: Ð, ð. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@dircategory` *partediretório*
: Especifica uma parte do menu do diretório do Info onde a entrada desse arquivo deveria ficar. Veja Seção 21.2.4 [Instalando Entradas de Diretório], Página 190.

`@direntry`
: Inicia a entrada do menu do diretório do Info para esse arquivo. Emparelha com `@end direntry`. Veja Seção 21.2.4 [Instalando Entradas de Diretório], Página 190.

`@display` Inicia um tipo de exemplo. Como `@example` (recua texto, não preenche), mas não seleciona uma nova fonte. Emparelha com `@end display`. Veja Seção 8.7 [`@display`], Página 70.

`@dmn{`*dimensão*`}`
: Formata uma unidade de medida, como em 12 pt. Faz com que TEX insira um espaço fino antes de *dimensão*. Sem efeito no Info. Veja Seção 12.3.5 [`@dmn`], Página 99.

`@docbook` Entra em Docbook completamente. Emparelha com `@end docbook`. Veja Seção 16.3 [Comandos do Formatador Bruto], Página 130.

`@documentdescription`
: Define o texto de descrição do documento, incluído na saída HTML. Emparelha com `@end documentdescription`. Veja Seção 3.7.1 [`@documentdescription`], Página 25.

`@documentencoding` *codificação*
: Declara a codificação de entrada como *codificação*. Veja Seção 15.2 [`@documentencoding`], Página 126.

`@documentlanguage` *CC*
: Declara o idioma do documento como a abreviação ISO-639 de dois caracteres *CC*. Veja Seção 15.1 [`@documentlanguage`], Página 125.

`@dotaccent{`*c*`}`
: Gera um acento de ponto sobre o caractere *c*, como em ȯ. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@dotless{`*i-ou-j*`}`
: Gera i sem ponto (‘ı’) e j sem ponto (‘ȷ’). Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@dots{}` Gera uma reticência, ‘`...`’. Veja Seção 12.8.4 [`@dots`], Página 104.

`@email{`*endereço*`[, `*texto-exibido*`]}`
: Indica um endereço eletrônico de mensagens. Veja Seção 7.1.16 [`@email`], Página 63.

`@emph{`*texto*`}`
: Enfatiza *texto*, usando *itálico*, onde possível e cercando com asteriscos no Info. Veja Seção 7.2 [Enfatizando Texto], Página 64.

`@end` *ambiente*
: Termina *ambiente*, como em ‘`@end example`’. Veja [Comandos @], Página 9.

`@enddots{}`
: Gera reticências de fim de frase, como esta: ... Veja Seção 12.8.4 [`@dots`], Página 104.

`@enumerate [`*número-ou-letra*`]`
: Inicia uma lista numerada, usando `@item` para cada entrada. Opcionalmente, inicia a lista com *número-ou-letra*. Emparelha com `@end enumerate`. Veja Seção 9.3 [`@enumerate`], Página 77.

`@env{`*variável-ambiente*`}`
: Indica um nome de variável de ambiente, como `CAMINHO`. Veja Seção 7.1.8 [`@env`], Página 61.

`@equiv{}` Indica para o(a) leitor(a) a equivalência exata de duas formas com um glifo: ‘≡’. Veja Seção 12.9.6 [`@equiv`], Página 107.

`@error{}` Indica para o(a) leitor(a) com um glifo que o texto seguinte é uma mensagem de erro: ‘ error ’. Veja Seção 12.9.5 [`@error`], Página 107.

`@errormsg{`*mensagem*`}`
: Informa *mensagem* como um erro para erro padrão e sai sem sucesso. Comandos do Texinfo dentro de *mensagem* são expandidos para texto simples. Veja Capítulo 16 [Condicionais], Página 128, e Seção 17.6 [Processadores Externos de Macro], Página 143.

`@euro{}` Gera o símbolo da moeda Euro. Veja Seção 12.8.6 [`@euro`], Página 104.

`@evenfooting [`*esquerda*`] @| [`*centro*`] @| [`*direita*`]`
`@evenheading [`*esquerda*`] @| [`*centro*`] @| [`*direita*`]`
: Especifica rodapés de página, respectivamente cabeçalhos, para páginas pares (à esquerda). Veja Seção E.4 [Como Fazer Teus Próprios Cabeçalhos], Página 249.

`@everyfooting [esquerda] @| [centro] @| [direita]`
`@everyheading [esquerda] @| [centro] @| [direita]`
: Especifica rodapés de página, respectivamente cabeçalhos, para cada página. Não relevante para Info. Veja Seção E.4 [Como Fazer Teus Próprios Cabeçalhos], Página 249.

`@example` Inicia um exemplo. Recua o texto, não preenche e seleciona fonte de largura fixa. Emparelha com `@end example`. Veja Seção 8.4 [`@example`], Página 68.

`@exampleindent recuo`
: Recua ambientes semelhantes a exemplos por *recuo* número de espaços (talvez 0). Veja Seção 3.7.6 [`@exampleindent`], Página 28.

`@exclamdown{}`
: Gere um ponto de exclamação invertido. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@exdent linha-do-texto`
: Remove qualquer recuo que uma linha possa ter. Veja Seção 8.9 [`@exdent`], Página 70.

`@expansion{}`
: Indica o resultado de uma expansão de macro para o(a) leitor(a) com um glifo especial: '↦'. Veja Seção 12.9.3 [`@expansion`], Página 106.

`@file{nomearquivo}`
: Destaca o nome de um arquivo, buffer, nó, diretório, etc. Veja Seção 7.1.9 [`@file`], Página 61.

`@finalout`
: Impede que o TEX imprima grandes retângulos pretos de aviso ao lado de linhas muito largas. Veja Seção 19.10 [hboxes lotados], Página 158.

`@findex entrada`
: Adiciona *entrada* ao índice de funções. Veja Seção 11.3 [Definindo as Entradas de um Índice], Página 90.

`@firstparagraphindent palavra`
: Controla o recuo do primeiro parágrafo depois dos cabeçalhos de seção de acordo com *palavra*, uma de 'none' ou 'insert'. Veja Seção 3.7.5 [`@firstparagraphindent`], Página 27.

`@float` Ambiente para definir material flutuante. Emparelha com `@end float`. Veja Seção 10.1 [Flutuações], Página 82.

`@flushleft`
`@flushright`
: Não preenche texto; justifica cada linha à esquerda (direita), deixando a extremidade direita (esquerda) irregular. Deixa a fonte como está. Emparelha com `@end flushleft` (`@end flushright`). Veja Seção 8.10 [`@flushleft @flushright`], Página 71.

`@fonttextsize 10-11`
: Muda o tamanho da fonte do corpo principal na saída do TEX. Veja Seção 7.2.3 [Fontes], Página 65.

`@footnote{texto-da-notaderodapé}`
: Insere uma nota de rodapé. O texto da nota de rodapé é impresso na parte inferior da página pelo TEX; Info pode formatar ou no estilo de nó 'End' ou nó 'Separate'. Veja Seção 10.3 [Notas de Rodapé], Página 86.

`@footnotestyle` *estilo*
: Especifica um estilo de nota de rodapé do arquivo do Info, ou '`end`' para o estilo do nó final ou '`separate`' para o estilo do nó separado. Veja Seção 10.3 [Notas de Rodapé], Página 86.

`@format`
: Inicia um tipo de exemplo. Como `@display`, mas não recua. Emparelha com `@end format`. Veja Seção 8.4 [`@example`], Página 68.

`@frenchspacing` *ligado-desligado*
: Controla o espaçamento depois da pontuação. Veja Seção 12.3.4 [`@frenchspacing`], Página 99.

`@ftable` *comando-formatação*
: Inicia uma tabela de duas colunas, usando `@item` para cada entrada. Insere automaticamente cada um dos itens da primeira coluna no índice de funções. Emparelha com `@end ftable`. O mesmo que `@table`, exceto pela indexação. Veja Seção 9.4.2 [`@ftable @vtable`], Página 79.

`@geq{}`
: Gera um sinal de maior que ou igual a, '≥'. Veja Seção 12.8.10 [`@geq @leq`], Página 105.

`@group`
: Desautoriza quebras de página dentro do texto seguinte. Emparelha com `@end group`. Ignorado no Info. Veja Seção 13.9 [`@group`], Página 112.

`@guillemetleft{}`
`@guillemetright{}`
`@guillemotleft{}`
`@guillemotright{}`
`@guilsinglleft{}`
`@guilsinglright{}`
: Aspas angulares duplas e simples: « » ‹ ›. `@guillemotleft` e `@guillemotright` são sinônimos de `@guillemetleft` e `@guillemetright`. Veja Seção 12.5 [Inserindo Aspas], Página 101.

`@H{`*c*`}`
: Gera o acento de trema húngaro longo sobre *c*, como em ő.

`@hashchar{}`
: Insere um caractere cerquilha '#'; necessário somente quando uma cerquilha literal introduziria a diretiva `#line`. Veja Seção 12.1.5 [Inserindo um Símbolo Cerquilha], Página 96, e Seção 17.6 [Processadores Externos de Macro], Página 143.

`@heading` *título*
: Imprime um título não numerado, como uma seção, mas omite da tabela de conteúdo. No Info, o título é sublinhado com sinais de igual. Veja Seção 5.7 [`@unnumberedsec @appendixsec @heading`], Página 41.

`@headings` *ligado-desligado-simples-duplo*
: Ativa ou desativa os títulos de página e (ou) especifica títulos de página de um ou de dois lados para impressão. Veja Seção 3.7.3 [`@headings`], Página 26.

`@headitem`
: Inicia uma linha de cabeçalho em uma multi tabela. Veja Seção 9.5.2 [Linhas de Multi Tabelas], Página 80.

`@headitemfont{`*texto*`}`
: Configura *texto* na fonte usada para linhas de cabeçalho de multi tabelas; útil principalmente em modelos de multi tabelas. Veja Seção 9.5.2 [Linhas de Multi Tabelas], Página 80.

`@html` Entra em HTML completamente. Emparelha com `@end html`. Veja Seção 16.3 [Comandos do Formatador Bruto], Página 130.

`@hyphenation{`*palavras hi-fen-a-das words*`}`
Define explicitamente os pontos de hifenização. Veja Seção 13.3 [`@- @hyphenation`], Página 111.

`@i{`*texto*`}`
Configura *texto* em uma fonte *itálica*. Sem efeito no Info. Veja Seção 7.2.3 [Fontes], Página 65.

`@ifclear` *variáveltexinfo*
Se a variável do Texinfo *variáveltexinfo* não estiver configurada, formata o texto seguinte. Emparelha com `@end ifclear`. Veja Seção 16.5 [`@set @clear @value`], Página 132.

`@ifcommanddefined` *comandotexinfo*
`@ifcommandnotdefined` *comandotexinfo*
Se o código do Texinfo '`@`*comandotexinfo*' (não) estiver definido, formata o texto a seguir. Emparelha com o correspondente `@end ifcommand...`. Veja Seção 16.6 [Testes para Comandos do Texinfo], Página 135.

`@ifdocbook`
`@ifhtml`
`@ifinfo` Inicia o texto que aparecerá somente no formato de saída fornecido. A saída do `@ifinfo` aparece tanto na saída do Info quanto (para compatibilidade histórica) em texto simples. Emparelha com `@end ifdocbook`, respectivamente `@end ifhtml`, respectivamente `@end ifinfo`. Veja Capítulo 16 [Condicionais], Página 128.

`@ifnotdocbook`
`@ifnothtml`
`@ifnotplaintext`
`@ifnottex`
`@ifnotxml`
Inicia o texto a ser ignorado em um formato de saída, mas não nos outros. Texto do `@ifnothtml` é omitido da saída HTML, etc. Emparelha com o `@end ifnot`*format* correspondente. Veja Capítulo 16 [Condicionais], Página 128.

`@ifnotinfo`
Inicia o texto a aparecer em uma saída diferente de Info e (para compatibilidade histórica) texto simples. Emparelha com `@end ifnotinfo`. Veja Capítulo 16 [Condicionais], Página 128.

`@ifplaintext`
Inicia o texto que aparecerá somente na saída de texto simples. Emparelha com `@end ifplaintext`. Veja Capítulo 16 [Condicionais], Página 128.

`@ifset` *variáveltexinfo*
Se a variável do Texinfo *variáveltexinfo* estiver configurada, formata o texto seguinte. Emparelha com `@end ifset`. Veja Seção 16.5 [`@set @clear @value`], Página 132.

`@iftex` Inicia o texto para aparecer somente na saída do TEX. Emparelha com `@end iftex`. Veja Capítulo 16 [Texto Visível Condicionalmente], Página 128.

`@ifxml` Inicia o texto que aparecerá somente na saída XML. Emparelha com `@end ifxml`. Veja Capítulo 16 [Condicionais], Página 128.

`@ignore` Inicia um texto que não aparecerá em nenhuma saída. Emparelha com `@end ignore`. Veja Seção 2.2 [Comentários e Texto Ignorado], Página 10.

`@image{`*nomearquivo*`, [`*largura*`], [`*altura*`], [`*alternativo*`], [`*extensão*`]}`
: Inclui imagem gráfica em *nomearquivo* externo dimensionado para *largura* e (ou) *altura* fornecidos, usando texto *alternativo* e procurando por '*nomearquivo*`.`*extensão*' em HTML. Veja Seção 10.2 [Imagens], Página 84.

`@include` *nomearquivo*
: Lê o conteúdo do arquivo fonte do Texinfo *nomearquivo*. Veja Capítulo 18 [Arquivos de Inclusão], Página 146.

`@indent` Insere recuo de parágrafo. Veja Seção 8.13 [`@indent`], Página 72.

`@indentedblock`
: Recua um bloco de texto arbitrário à esquerda. Emparelha com `@end indentedblock`. Veja Seção 8.3 [`@indentedblock`], Página 68.

`@indicateurl{`*urlindicação*`}`
: Indica o texto que é um localizador uniforme de recursos para a World Wide Web. Veja Seção 7.1.15 [`@indicateurl`], Página 63.

`@inforef{`*nome-nó*`, [`*nome-entrada*`], `*nome-arquivo-info*`}`
: Faz uma referência cruzada para um arquivo do Info para o qual não existe manual impresso. Veja Seção 6.9 [`@inforef`], Página 52.

`@inlinefmt{`*formato*`, `*texto*`}`
: Insere *texto* somente se o formato de saída for *formato*. Veja Seção 16.4 [Condicionais Inline], Página 131.

`@inlinefmtifelse{`*formato*`, `*texto*`, `*texto-senão*`}`
: Insere *texto* se o formato de saída for *formato*, caso contrário *texto-senão*.

`@inlineifclear{`*variável*`, `*texto*`}`
`@inlineifset{`*variável*`, `*texto*`}`
: Insere *texto* somente se a variável do Texinfo *variável* (não) estiver configurada.

`@inlineraw{`*formato*`, `*texto-bruto*`}`
: Insere *texto* como em uma condicional bruta, somente se o formato de saída for *formato*.

`\input` *arquivo-definições-macro*
: Usa o arquivo de definições de macro especificado. Esse comando é usado somente na primeira linha de um arquivo do Texinfo para fazer com que o TEX faça uso do arquivo de definições de macro `texinfo`. A `\` em `\input` é usada, em vez de um `@`, porque o TEX não reconhece `@` até depois de ter lido o arquivo de definições. Veja Seção 3.2 [Cabeçalho do Arquivo do Texinfo], Página 15.

`@insertcopying`
: Insere o texto definido anteriormente com o ambiente `@copying`. Veja Seção 3.3.2 [`@insertcopying`], Página 18.

`@item` Indica o início de um parágrafo marcado para `@itemize` e `@enumerate`; indica o início do texto de uma entrada de primeira coluna para `@table`, `@ftable` e `@vtable`. Veja Capítulo 9 [Listas e Tabelas], Página 75.

`@itemize` *marca-caractere-gerador-ou-comando*
: Inicia uma lista não ordenada: parágrafos recuados com uma marca, como `@bullet`, dentro da margem esquerda no início de cada item. Emparelha com `@end itemize`. Veja Seção 9.2 [`@itemize`], Página 75.

`@itemx` Como `@item`, mas não gera espaço vertical extra acima do texto do item. Assim, quando vários itens tiverem a mesma descrição, usa `@item` para o primeiro e `@itemx` para os outros. Veja Seção 9.4.3 [`@itemx`], Página 79.

`@kbd{`*`caracteres-teclado`*`}`
: Indica caracteres de entrada a serem digitados por usuários(as). Veja Seção 7.1.3 [`@kbd`], Página 58.

`@kbdinputstyle` *`estilo`*
: Especifica quando `@kbd` deveria usar uma fonte diferente de `@code` de acordo com *estilo*: `code`, `distinct`, `example`. Veja Seção 7.1.3 [`@kbd`], Página 58.

`@key{`*`nome-tecla`*`}`
: Indica o nome de uma tecla em um teclado. Veja Seção 7.1.4 [`@key`], Página 59.

`@kindex` *`entrada`*
: Adiciona *entrada* ao índice de chaves. Veja Seção 11.3 [Definindo as Entradas de um Índice], Página 90.

`@L{}`
`@l{}`
: Gera as letras L suprimidas, maiúsculas e minúsculas, em polonês, respectivamente: Ł, ł.

`@LaTeX{}`
: Gera o logotipo do LaTeX. Veja Seção 12.8.1 [`@TeX` `@LaTeX`], Página 103.

`@leq{}`
: Gera um sinal de menor que ou igual a, '≤'. Veja Seção 12.8.10 [`@geq` `@leq`], Página 105.

`@lisp`
: Inicia um exemplo de código Lisp. Recua o texto, não preenche e seleciona fonte de largura fixa. Emparelha com `@end lisp`. Veja Seção 8.6 [`@lisp`], Página 70.

`@listoffloats`
: Produz uma listagem de `@float`s, semelhante a um sumário. Veja Seção 10.1.3 [`@listoffloats`], Página 83.

`@lowersections`
: Muda capítulos subsequentes para seções, seções para subseções e assim por diante. Veja Seção 5.12 [`@raisesections` e `@lowersections`], Página 44.

`@macro` *`nomemacro`* `{`*`parâmetros`*`}`
: Define um novo comando do Texinfo `@`*`nomemacro`*`{`*`parâmetros`*`}`. Emparelha com `@end macro`. Veja Seção 17.1 [Definindo Macros], Página 137.

`@majorheading` *`título`*
: Imprime um título não numerado, como um capítulo, mas omite do sumário. Isso gera mais espaço em branco vertical antes do título que o comando `@chapheading`. Veja Seção 5.5 [`@majorheading` `@chapheading`], Página 41.

`@math{`*`expressão-matemática`*`}`
: Formata uma expressão matemática. Veja Seção 12.7 [Inserindo Fórmulas Matemáticas], Página 102.

`@menu`
: Marca o início de um menu de nós. Nenhum efeito em um manual impresso. Emparelha com `@end menu`. Veja Seção 4.9 [Menus], Página 36.

`@minus{}`
: Gera um sinal de menos, '−'. Veja Seção 12.8.9 [`@minus`], Página 105.

`@multitable` *`coluna-jargura-especificações`*
: Inicia uma tabela com várias colunas. Inicie cada linha com `@item` ou `@headitem` e separe as colunas com `@tab`. Emparelha com `@end multitable`. Veja Seção 9.5.1 [Larguras de Colunas Multi Tabelas], Página 80.

`@need` *`n`*
: Inicia uma nova página em um manual impresso se menos que *n* mils (milésimos de polegada) restarem na página atual. Veja Seção 13.10 [`@need`], Página 113.

`@node` *nome, próximo, anterior, acima*
: Inicia um novo nó. Veja Seção 4.3 [Escrevendo um Nó], Página 30.

`@noindent`
: Evita que o texto seja recuado como se fosse um novo parágrafo. Veja Seção 8.12 [@noindent], Página 72.

`@novalidate`
: Suprime a validação de referências de nó e omite a criação de arquivos auxiliares com o TEX. Use antes de quaisquer comandos de seccionamento ou de referência cruzada. Veja Seção 20.4 [Validação de Ponteiro], Página 169.

`@O{}`
`@o{}` Gera as letras O com barra, maiúsculas e minúsculas, respectivamente: Ø, ø.

`@oddfooting [`*esquerda*`] @| [`*centro*`] @| [`*direita*`]`
`@oddheading [`*esquerda*`] @| [`*centro*`] @| [`*direita*`]`
: Especifica os rodapés das páginas, respectivamente títulos, para páginas ímpares (à direita). Veja Seção E.4 [Como Fazer Teus Próprios Cabeçalhos], Página 249.

`@OE{}`
`@oe{}` Gera as ligaduras OE maiúsculas e minúsculas, respectivamente: Œ, œ. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@ogonek{`*c*`}`
: Gera um diacrítico ogonek sob o próximo caractere, como em ą. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@option{`*nome-opção*`}`
: Indica uma opção de linha de comando, como `-l` ou `--help`. Veja Seção 7.1.11 [@option], Página 62.

`@ordf{}`
`@ordm{}` Gera os ordinais espanhóis femininos e masculinos, respectivamente: ª, º. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@page` Inicia uma nova página em um manual impresso. Nenhum efeito no Info. Veja Seção 13.8 [@page], Página 112.

`@pagesizes [`*largura*`][, `*altura*`]`
: Muda dimensões da página. Veja [pagesizes], Página 160.

`@paragraphindent` *recuo*
: Recua parágrafos por *recuo* número de espaços (talvez 0); preserva o recuo do arquivo fonte se *recuo* for `asis`. Veja Seção 3.7.4 [@paragraphindent], Página 27.

`@part` *título*
: Inicia um grupo de capítulos ou anexos; incluídos no sumário e produz uma página própria na saída impressa. Veja Seção 5.11 [@part], Página 43.

`@pindex` *entrada*
: Adiciona *entrada* ao índice de programas. Veja Seção 11.3 [Definindo as Entradas de um Índice], Página 90.

`@point{}` Indica a posição do ponto em um buffer para o(a) leitor(a) com um glifo: '⋆'. Veja Seção 12.9.7 [@point], Página 107.

`@pounds{}`
: Gera o símbolo da moeda libras esterlinas. Veja Seção 12.8.7 [@pounds], Página 105.

`@print{}` Indica a saída impressa para o(a) leitor(a) com um glifo: ' ⊣ '. Veja Seção 12.9.4 [@print], Página 106.

`@printindex` *nome-índice*
: Gera o índice alfabético para *nome-índice* (usando duas colunas em um manual impresso). Veja Seção 11.4 [Imprimindo Índices e Menus], Página 91.

`@pxref{`*nó*`, [`*entrada*`], [`*título-nó*`], [`*arquivo-info*`], [`*manual*`]}`
: Faz uma referência que comece com um 'Veja' minúsculo em um manual impresso. Use somente entre parênteses. Somente o primeiro argumento é obrigatório. Veja Seção 6.7 [`@pxref`], Página 50.

`@questiondown{}`
: Gera um ponto de interrogação invertido. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@quotation`
: Estreita as margens para indicar texto que seja citado a partir de outro trabalho. Recebe argumento opcional especificando texto de prefixo, por exemplo, um nome de autor(a). Emparelha com `@end quotation`. Veja Seção 8.2 [`@quotation`], Página 67.

`@quotedblleft{}`
`@quotedblright{}`
`@quoteleft{}`
`@quoteright{}`
`@quotedblbase{}`
`@quotesinglbase{}`
: Produz várias aspas: “ ” ‘ ’ „ ,. Veja Seção 12.5 [Inserindo Aspas], Página 101.

`@r{`*texto*`}`
: Configura *texto* na fonte regular romana. Sem efeito no Info. Veja Seção 7.2.3 [Fontes], Página 65.

`@raggedright`
: Preenche texto; justifica à esquerda cada linha, deixando a extremidade direita irregular. Deixa a fonte como está. Emparelha com `@end raggedright`. Sem efeito no Info. Veja Seção 8.11 [`@raggedright`], Página 71.

`@raisesections`
: Muda seções subsequentes para capítulos, subseções para seções e assim por diante. Veja Seção 5.12 [Elevar/rebaixar seções], Página 44.

`@ref{`*nó*`, [`*entrada*`], [`*título-nó*`], [`*arquivo-info*`], [`*manual*`]}`
: Faz uma referência simples que não comece com nenhum texto especial. Siga o comando com um sinal de pontuação. Somente o primeiro argumento é obrigatório. Veja Seção 6.6 [`@ref`], Página 50.

`@refill`
: Esse comando costumava preencher e recuar o parágrafo depois que todo o outro processamento tivesse sido feito. Ele não mais é necessário, já que todos os formatadores agora preenchem automaticamente conforme necessário, mas você ainda pode vê-lo no fonte para alguns manuais, pois não faz mal.

`@registeredsymbol{}`
: Gera o símbolo jurídico ®. Veja Seção 12.8.3 [`@registeredsymbol`], Página 104.

`@result{}`
: Indica o resultado de uma expressão para o(a) leitor(a) com um glifo especial: '⇒'. Veja Seção 12.9.2 [`@result`], Página 106.

`@ringaccent{`*c*`}`
: Gera um acento de anel sobre o próximo caractere, como em ů. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@samp{`*texto*`}`
: Indica um exemplo literal de uma sequência de caracteres, em geral. Aspado na saída do Info. Veja Seção 7.1.5 [`@samp`], Página 59.

`@sansserif{`*texto*`}`
: Configura *texto* em uma fonte sans serifa se possível. Sem efeito no Info. Veja Seção 7.2.3 [Fontes], Página 65.

`@sc{`*texto*`}`
: Configura *texto* em fonte de versaletes na saída impressa e em caixa alta no Info. Veja Seção 7.2.2 [Versaletes], Página 64.

`@section` *título*
: Inicia uma seção dentro de um capítulo. O título da seção aparece no sumário. Em Info, o título é sublinhado com sinais de igual. Dentro de `@chapter` e `@appendix`, o título da seção é numerado; dentro de `@unnumbered`, a seção não é numerada. Veja Seção 5.6 [`@section`], Página 41.

`@set` *variáveltexinfo* `[`*string*`]`
: Define a variável do Texinfo *variáveltexinfo*, opcionalmente para o valor *string*. Veja Seção 16.5 [`@set @clear @value`], Página 132.

`@setchapternewpage` *ligado-desligado-ímpar*
: Especifica se os capítulos iniciam em novas páginas e, em caso afirmativo, se em novas páginas ímpares (à direita). Veja Seção 3.7.2 [`@setchapternewpage`], Página 25.

`@setcontentsaftertitlepage`
: Coloca o sumário depois do '`@end titlepage`', mesmo que o comando `@contents` esteja no final. Veja Seção 3.5 [Conteúdo], Página 22.

`@setfilename` *nome-arquivo-info*
: Fornece um nome a ser usado para os arquivos de saída. Esse comando é ignorado para formatação de TEX. Veja Seção 3.2.3 [`@setfilename`], Página 16.

`@setshortcontentsaftertitlepage`
: Coloca o sumário curto depois do comando '`@end titlepage`', mesmo que o comando `@shortcontents` esteja no final. Veja Seção 3.5 [Conteúdo], Página 22.

`@settitle` *título*
: Especifica o título para cabeçalhos de página em um manual impresso e o título padrão de documento para '`<head>`' do HTML. Veja Seção 3.2.4 [`@settitle`], Página 17.

`@shortcaption`
: Define a legenda curta para um `@float`. Veja Seção 10.1.2 [`@caption @shortcaption`], Página 83.

`@shortcontents`
: Imprime um sumário curto, com entradas de nível de capítulo somente. Não é relevante para Info, que usa menus em vez de sumários. Veja Seção 3.5 [Gerando Um Sumário], Página 22.

`@shorttitlepage` *título*
: Gera uma página mínima de título. Veja Seção 3.4.1 [`@titlepage`], Página 19.

`@slanted{`*texto*`}`
: Configura *texto* em uma fonte *inclinada* se possível. Sem efeito no Info. Veja Seção 7.2.3 [Fontes], Página 65.

`@smallbook`
: Faz com que o TEX produza um manual impresso em um formato de 7 por 9,25 polegadas em vez do formato regular de 8,5 por 11 polegadas. Veja Seção 19.11 [`@smallbook`], Página 159. Veja-se também Seção 8.15 [`@small...`], Página 73.

`@smalldisplay`
: Inicia um tipo de exemplo. Como `@display`, mas usa um tamanho de fonte menor onde possível. Emparelha com `@end smalldisplay`. Veja Seção 8.15 [`@small...`], Página 73.

`@smallexample`
: Inicia um exemplo. Como `@example`, mas usa um tamanho de fonte menor onde possível. Emparelha com `@end smallexample`. Veja Seção 8.15 [`@small...`], Página 73.

`@smallformat`
: Inicia um tipo de exemplo. Como `@format`, mas usa um tamanho de fonte menor onde possível. Emparelha com `@end smallformat`. Veja Seção 8.15 [`@small...`], Página 73.

`@smallindentedblock`
: Como `@indentedblock`, mas usa um tamanho de fonte menor onde possível. Emparelha com `@end smallindentedblock`. Veja Seção 8.15 [`@small...`], Página 73.

`@smalllisp`
: Inicia um exemplo de código Lisp. O mesmo que `@smallexample`. Emparelha com `@end smalllisp`. Veja Seção 8.15 [`@small...`], Página 73.

`@smallquotation`
: Como `@quotation`, mas usa um tamanho de fonte menor onde possível. Emparelha com `@end smallquotation`. Veja Seção 8.15 [`@small...`], Página 73.

`@sortas {`*`chave`*`}`
: Usado nos argumentos para comandos de indexação para fornecer uma string pela qual a entrada do índice deveria ser ordenada. Veja Seção 11.2 [Comandos de Indexação], Página 90.

`@sp` *`n`*
: Ignora *n* linhas em branco. Veja Seção 13.7 [`@sp`], Página 112.

`@ss{}`
: Gera a letra alemã S sustenido es-zet, ß. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@strong {`*`texto`*`}`
: Enfatiza *texto* com mais força que `@emph`, usando **negrito** onde possível; cercado entre asteriscos no Info. Veja [Enfatizando Texto], Página 64.

`@sub {`*`texto`*`}`
: Configura *texto* como um subscrito. Veja Seção 12.6 [Inserindo Subscritos e Sobrescritos], Página 102.

`@subheading` *`título`*
: Imprime um título não numerado, como uma subseção, mas omite do sumário de um manual impresso. No Info, o título é sublinhado com hifens. Veja Seção 5.9 [`@unnumberedsubsec @appendixsubsec @subheading`], Página 42.

`@subsection` *`título`*
: Inicia uma subseção dentro de uma seção. O título da subseção aparece no sumário. No Info, o título é sublinhado com hifens. Mesma numeração dependente do contexto que `@section`. Veja Seção 5.8 [`@subsection`], Página 42.

`@subsubheading` *título*
: Imprime um título não numerado, como uma subsubseção, mas omite do sumário de um manual impresso. No Info, o título é sublinhado com pontos. Veja Seção 5.10 [`@subsubsection`], Página 42.

`@subsubsection` *título*
: Inicia uma subsubseção dentro de uma subseção. O título da subsubseção aparece no sumário. No Info, o título é sublinhado com pontos. Mesma numeração dependente do contexto que `@section`. Veja Seção 5.10 [`@subsubsection`], Página 42.

`@subtitle` *título*
: Em um manual impresso, configura um subtítulo em uma fonte de tamanho normal alinhada ao lado direito da página. Não é relevante para Info, que não tem páginas de título. Veja Seção 3.4.3 [`@title @subtitle @author`], Página 20.

`@summarycontents`
: Imprime um sumário curto. Sinônimo para `@shortcontents`. Veja Seção 3.5 [Gerando Um Sumário], Página 22.

`@sup {`*texto*`}`
: Configura *texto* como um sobrescrito. Veja Seção 12.6 [Inserindo Subscritos e Sobrescritos], Página 102.

`@syncodeindex` *de-índice para-índice*
: Mescla o índice nomeado no primeiro argumento no índice nomeado no segundo argumento, formatando as entradas originárias do primeiro índice com `@code`. Veja Seção 11.5 [Combinando Índices], Página 92.

`@synindex` *de-índice para-índice*
: Mescla o índice nomeado no primeiro argumento no índice nomeado no segundo argumento. Não muda a fonte das entradas de *de-índice*. Veja Seção 11.5 [Combinando Índices], Página 92.

`@t{`*texto*`}`
: Configura *texto* em uma fonte de `largura-fixa`, tipo máquina de escrever. Sem efeito no Info. Veja Seção 7.2.3 [Fontes], Página 65.

`@tab` Separa colunas em uma linha de uma multitabela. Veja Seção 9.5.2 [Linhas de Multi Tabelas], Página 80.

`@table` *comando-formatação*
: Inicia uma tabela de duas colunas (lista de descrição), usando `@item` para cada entrada. Escreve cada entrada da primeira coluna na mesma linha que `@item`. As entradas da primeira coluna são impressas na fonte resultante proveniente de *comando-formatação*. Emparelha com `@end table`. Veja Seção 9.4 [Fazendo Uma Tabela de Duas Colunas], Página 78. Veja-se também Seção 9.4.2 [`@ftable @vtable`], Página 79, e Seção 9.4.3 [`@itemx`], Página 79.

`@TeX{}` Gera o logotipo do TEX. Veja Seção 12.8.1 [`@TeX @LaTeX`], Página 103.

`@tex` Entra no TEX completamente. Emparelha com `@end tex`. Veja Seção 16.3 [Comandos do Formatador Bruto], Página 130.

`@textdegree{}`
: Gera o símbolo de Graus. Veja Seção 12.8.8 [`@textdegree`], Página 105.

`@thischapter`
`@thischaptername`
`@thischapternum`
`@thisfile`
`@thispage`
`@thistitle`

Permitido somente em um título ou rodapé. Representa, respectivamente, o número e o nome do capítulo atual (no formato 'Capítulo 1: Título'), somente o nome do capítulo atual, somente o número do capítulo atual, o nome do arquivo, o número da página atual e o título do documento, respectivamente. Veja Seção E.4 [Como Fazer Teus Próprios Cabeçalhos], Página 249.

`@TH{}`
`@th{}` Gera a letra maiúscula e minúscula islandesa thorn, respectivamente: Þ, þ. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@tie{}` Gera um espaço normal entre palavras no qual uma quebra de linha não é permitida. Veja Seção 13.6 [`@tie`], Página 112.

`@tieaccent{`*cc*`}`

Gera um acento de ligação sobre os próximos dois caracteres *cc*, como em 'o͡o'. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@tindex` *entrada*

Adiciona *entrada* ao índice de tipos de dados. Veja Seção 11.3 [Definindo as Entradas de um Índice], Página 90.

`@title` *título*

Em um manual impresso, configura um título alinhado ao lado esquerdo da página, em uma fonte maior que a normal e sublinhada com uma régua preta. Não é relevante para Info, que não tem páginas de título. Veja Seção 3.4.3 [`@title @subtitle @author`], Página 20.

`@titlefont{`*texto*`}`

Em um manual impresso, imprime *texto* em uma fonte maior que a normal. Veja Seção 3.4.2 [`@titlefont @center @sp`], Página 19.

`@titlepage`

Inicia a página de título. Escreve o comando em uma linha própria, emparelhado com `@end titlepage`. Nada entre `@titlepage` e `@end titlepage` aparece no Info. Veja Seção 3.4.1 [`@titlepage`], Página 19.

`@today{}` Insere a data atual, no estilo '1 de janeiro de 1900'. Veja Seção E.4 [Como Fazer Teus Próprios Cabeçalhos], Página 249.

`@top` *título*

Marca o `@node` mais alto no arquivo, que precisa ser definido na linha imediatamente precedente ao comando `@top`. O título é formatado como um cabeçalho de nível de capítulo. O nó superior inteiro, incluindo as linhas `@node` e `@top`, normalmente são cercadas com `@ifnottex ... @end ifnottex`. No TEX e `texinfo-format-buffer`, o comando `@top` é meramente um sinônimo para `@unnumbered`. Veja Seção 4.8 [Criação de Ponteiros do `makeinfo`], Página 35.

`@U{`*hexadecimal*`}`

Produz uma representação do caractere Unicode U+*hexadecimal*. Veja Seção 12.10 [Inserindo Unicode], Página 108.

`@u{`*c*`}`
`@ubaraccent{`*c*`}`
`@udotaccent{`*c*`}`
: Gera um acento breve, um sublinhado ou um ponto, respectivamente, sobre ou sob o caractere *c*, como em ŏ, o̲, ọ. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@unmacro` *`nomemacro`*
: Desconfigura a macro `@`*`nomemacro`* se ela tiver sido definida. Veja Seção 17.1 [Definindo Macros], Página 137.

`@unnumbered` *`título`*
: Inicia um capítulo que aparece sem números de capítulo de qualquer tipo. O título aparece no sumário. No Info, o título é sublinhado com asteriscos. Veja Seção 5.4 [`@unnumbered @appendix`], Página 40.

`@unnumberedsec` *`título`*
: Inicia uma seção que aparece sem números de seção de qualquer tipo. O título aparece no sumário de um manual impresso. No Info, o título é sublinhado com sinais de igual. Veja Seção 5.7 [`@unnumberedsec @appendixsec @heading`], Página 41.

`@unnumberedsubsec` *`título`*
: Inicia uma subseção não numerada. O título aparece no sumário. No Info, o título é sublinhado com hifens. Veja Seção 5.9 [`@unnumberedsubsec @appendixsubsec @subheading`], Página 42.

`@unnumberedsubsubsec` *`título`*
: Inicia uma subsubseção não numerada. O título aparece no sumário. No Info, o título é sublinhado com pontos. Veja Seção 5.10 [`@subsubsection`], Página 42.

`@uref{`*`url`*`[,` *`texto-exibido`*`][,` *`substituição`*`}`
`@url{`*`url`*`[,` *`texto-exibido`*`][,` *`substituição`*`}`
: Define uma referência cruzada para um localizador externo uniforme de recursos, por exemplo, para a World Wide Web. Veja Seção 6.10 [`@url`], Página 52.

`@urefbreakstyle` *`estilo`*
: Especifica como `@uref`/`@url` deveria quebrar em caracteres especiais: `after`, `before`, `none`. Veja Seção 6.10 [`@url`], Página 52.

`@v{`*c*`}` Gera acento caron sobre o caractere *c*, como em ŏ. Veja Seção 12.4 [Inserindo Acentos], Página 100.

`@validatemenus` *`ligado-desligado`*
: Controla se os menus podem ser gerados automaticamente. Veja Seção 4.9.1 [Escrevendo um Menu], Página 36.

`@value{`*`variáveltexinfo`*`}`
: Insere o valor, se existir, da variável do Texinfo *variáveltexinfo*, previamente definida por `@set`. Veja Seção 16.5 [`@set @clear @value`], Página 132.

`@var{`*`variável-metassintática`*`}`
: Destaca uma variável metassintática, que é algo que representa outro pedaço de texto. Veja Seção 7.1.7 [`@var`], Página 60.

`@verb{`*`delimitador literal delimitador`*`}`
: Emite *literal*, delimitada pelo caractere único *delimitador*, exatamente como está (na fonte de largura fixa), incluindo quaisquer espaços em branco ou caracteres especiais do Texinfo. Veja Seção 7.1.6 [`@verb`], Página 60.

`@verbatim`
: Emite o texto do ambiente exatamente como está (na fonte de largura fixa). Emparelha com `@end verbatim`. Veja Seção 8.5 [`@verbatim`], Página 69.

`@verbatiminclude` *nomearquivo*
: Emite o conteúdo de *nomearquivo* exatamente como está (na fonte de largura fixa). Veja Seção 18.5 [`@verbatiminclude`], Página 148.

`@vindex` *entrada*
: Adiciona *entrada* ao índice de variáveis. Veja Seção 11.3 [Definindo as Entradas de um Índice], Página 90.

`@vskip` *quantidade*
: Em um manual impresso, insere espaços em branco de forma a empurrar o texto no restante da página em direção ao final da página. Usado na formatação da página de direitos autorais com o argumento '`0pt plus 1filll`'. (Observe a grafia de '`filll`'). `@vskip` pode ser usado somente em contextos ignorados para Info. Veja Seção 3.4.4 [Direitos Autorais], Página 21.

`@vtable` *comando-formatação*
: Inicia uma tabela de duas colunas, usando `@item` para cada entrada. Insere automaticamente cada um dos itens da primeira coluna no índice de variáveis. Emparelha com `@end vtable`. O mesmo que `@table`, exceto pela indexação. Veja Seção 9.4.2 [`@ftable @vtable`], Página 79.

`@w{`*texto*`}`
: Proíbe quebras de linha em *texto*. Veja Seção 13.5 [`@w`], Página 111.

`@xml`
: Entra no XML completamente. Emparelha com `@end xml`. Veja Seção 16.3 [Comandos do Formatador Bruto], Página 130.

`@xref{`*nó*`, [`*entrada*`], [`*título-nó*`], [`*arquivo-info*`], [`*manual*`]}`
: Faz uma referência que comece com 'Veja' em um manual impresso. Siga o comando com um sinal de pontuação. Apenas o primeiro argumento é obrigatório. Veja Seção 6.4 [`@xref`], Página 47.

`@xrefautomaticsectiontitle` *ligado-desligado*
: Por padrão, use o título da seção em vez do nome do nó em referências cruzadas. Veja Seção 6.4.3 [Três Argumentos], Página 48.

A.3 Contextos de Comandos @

Aqui nós descrevemos aproximadamente quais comandos @ podem ser usados em quais contextos. Não é exaustivo nem pretende ser uma referência completa. Discrepâncias entre as informações aqui e as implementações do `makeinfo` ou do TEX provavelmente serão resolvidas em favor da implementação.

Por *texto geral* abaixo, nós queremos dizer qualquer coisa, exceto seccionamento e outros comandos de documento de nível externo, como `@section`, `@node` e `@setfilename`.

Os comandos condicionais `@c`, `@comment` e `@if ... @end if` podem aparecer em qualquer lugar (exceto os Condicionais, que ainda precisam estar em linhas separadas). `@caption` pode aparecer somente em `@float`, mas pode conter texto geral. O conteúdo de `@footnote` também.

Os comandos @ com chaves marcando texto (como `@strong`, `@sc`, `@asis`) podem conter comandos de formatador bruto, como `@html`, mas nenhum outro comando de bloco (outros comandos terminados por `@end`) e não podem ser divididos entre em parágrafos, mas podem conter texto geral.

Além da restrição do comando de bloco, em `@center`, `@exdent` e `@item` nas linhas `@table`, comandos @ que só fazem sentido em um parágrafo não são aceitos, como `@indent`.

Além do acima, os comandos de seccionamento não podem conter `@anchor`, `@footnote` ou `@verb`.

Além do acima, os comandos restantes (`@node`, `@anchor`, `@printindex`, `@ref`, `@math`, `@cindex`, `@url`, `@image` e assim por diante) não podem conter comandos de referência cruzada (`@ref`, `@xref`, `@pxref` e `@inforef`). Em uma última adição, `@shortcaption` pode aparecer somente dentro de `@float`.

Para informações precisas e completas, nós sugerimos consultar a suíte de teste nos fontes, que testa combinações exaustivamente.

Apêndice B Dicas e Sugestões

Aqui estão algumas dicas para escrever documentação do Texinfo:

- Escreva no presente, não no passado ou no futuro.
- Escreva ativamente! Por exemplo, escreva "Nós recomendamos que ..." em vez de "É recomendado que ...".
- Use 70 ou 72 como tua coluna de preenchimento. Linhas mais longas são difíceis de ler.
- Inclua um aviso de direitos autorais e de permissões de cópia.

Índice, Índice, Índice!

Escreva muitas entradas de índice, de maneiras diferentes. Leitores(as) gostam de índices; eles são úteis e convenientes.

Embora seja mais fácil escrever entradas de índice conforme você escreve o corpo do texto, algumas pessoas preferem escrever entradas depois. Em ambos os casos, escreva uma entrada antes do parágrafo ao qual ela se aplica. Dessa maneira, uma entrada de índice aponta para a primeira página de um parágrafo que é dividido entre páginas.

Aqui estão mais dicas relacionadas ao índice que nós consideramos valiosas:

- Escreva cada entrada de índice diferentemente, de forma que cada entrada se refira a um lugar diferente no documento.
- Escreva entradas de índice somente onde um tópico seja discutido significativamente. Por exemplo, não é útil indexar "informações de depuração" em um capítulo acerca de informes de defeitos. Alguém que queira saber acerca de informações de depuração certamente não as encontrará naquele capítulo.
- Capitalize consistentemente a primeira palavra de cada entrada do índice de conceitos, ou então use consistentemente letras minúsculas. Entradas concisas frequentemente pedem letras minúsculas; entradas mais longas pedem letras maiúsculas. Qualquer que seja a convenção de caixa que você usar, por favor, use uma ou a outra consistentemente! Misturar os dois estilos parece ruim.
- Sempre coloque em maiúsculas ou use letras maiúsculas para aquelas palavras em um índice para as quais isso for apropriado, como nomes de países ou siglas. Sempre use a caixa apropriada para nomes sensíveis a maiúsculas e minúsculas, como aqueles em C ou Lisp.
- Escreva os comandos de indexação que se refiram a uma seção inteira imediatamente depois do comando de seção e escreva os comandos de indexação que se refiram a um parágrafo antes desse parágrafo.

  No exemplo que segue, uma linha em branco vem depois da entrada de índice para "Pulando":

  ```
  @section O Cachorro e a Raposa
  @cindex Pular, em geral
  @cindex Pulando

  @cindex Cachorro, preguiçoso, pulou
  @cindex Cachorro preguiçoso, pulou
  @cindex Raposa, pula sobre o cachorro
  @cindex Raposa rápida salta sobre o cachorro
  A rápida raposa marrom salta sobre o cachorro preguiçoso.
  ```

  (Observe que o exemplo mostra entradas para o mesmo conceito que são escritas de maneiras diferentes—'`Cachorro preguiçoso`' e '`Cachorro, preguiçoso`'—de forma que os(as) leitores(as) consigam pesquisar o conceito de maneiras diferentes).

Linhas em Branco

- Insira uma linha em branco entre um comando de seccionamento e a primeira frase ou parágrafo seguinte, ou entre os comandos de indexação associados com o comando de seccionamento e a primeira frase ou parágrafo seguinte, como mostrado na dica acerca de indexação. Isso torna o fonte mais fácil de ler.
- Sempre insira uma linha em branco antes de um comando `@table` e depois de um comando `@end table`; mas nunca insira uma linha em branco depois de um comando `@table`.

  Por exemplo,

```
Tipos de raposa:

@table @samp
@item Rápida
Pula sobre cães preguiçosos.

@item Marrom
também pula sobre cães preguiçosos.
@end table

@noindent
Por outro lado, ...
```

  Insira linhas em branco antes e depois de `@itemize` ... `@end itemize` e `@enumerate` ... `@end enumerate` da mesma maneira.

Frases Completas

Frases completas são mais fáceis de ler que ...

- Escreva entradas em uma lista itemizada como sentenças completas; ou pelo menos, como frases completas. Expressões incompletas ... estranhas ... como esta.
- Escreva a frase ou sentença preliminar para uma lista ou tabela de vários itens como uma expressão completa. Não escreva "Você pode configurar:"; em vez disso, escreva "Você pode configurar estas variáveis:". A expressão anterior soa cortada.

Edições, Datas e Versões

Inclua números de edição, números de versão e datas no texto do `@copying` (para pessoas que leem o arquivo do Texinfo e para os direitos autorais nos arquivos de saída). Em seguida, use `@insertcopying` na seção `@titlepage` para pessoas que leem a saída impressa (veja Seção 2.4 [Amostra Curta], Página 11).

É mais fácil manusear essas informações de versão usando `@set` e `@value`. Veja Seção 16.5.4 [Exemplo de `@value`], Página 134, e Seção C.2 [Textos GNU de Amostra], Página 233.

Comandos de Definição

Os comandos de definição são `@deffn`, `@defun`, `@defmac` e similares, e te habilitam a escrever descrições em um formato uniforme.

- Escreva apenas um comando de definição para cada entidade que você definir com um comando de definição. O recurso de indexação automática cria uma entrada de índice que leva o(a) leitor(a) para a definição.
- Use `@table` ... `@end table` em um anexo que contenha um resumo de funções, não `@deffn` ou outros comandos de definição.

Capitalização

- Coloque "Texinfo" em maiúscula; é um nome. Não escreva '`x`' ou o '`i`' em maiúsculas.
- Coloque "Info" em maiúscula; é um nome.
- Escreva TEX usando o comando `@TeX{}`. Observe as letras maiúsculas '`T`' e '`X`'. Esse comando faz com que os formatadores tipografem o nome de acordo com os desejos de Donald Knuth, que escreveu o TEX. (Da mesma forma `@LaTeX{}` para LATEX).

Espaços

Não use espaços para formatar um arquivo do Texinfo, exceto dentro de `@example ... @end example` e outros ambientes e comandos literais.

Por exemplo, o TEX preenche o seguinte:

```
@kbd{C-x v}
@kbd{M-x vc-next-action}
   Realize a próxima operação lógica sobre arquivo controlado por versão
   correspondente ao buffer atual.
```

de forma que se pareça com isto:

> *C-x v M-x vc-next-action* Realize a próxima operação lógica sobre arquivo controlado por versão correspondente ao buffer atual.

Nesse caso, o texto deveria ser formatado com `@table`, `@item` e `@itemx` para criar uma tabela.

@code, @samp, @var e '---'

- Use `@code` em torno de símbolos da Lisp, incluindo nomes de comandos. Por exemplo,

  ```
  A função principal é @code{vc-next-action}, ...
  ```

- Evite colocar letras, como '`s`', imediatamente depois de um '`@code`'. Essas letras parecem ruins.
- Use `@var` em torno de meta-variáveis. Não escreva colchetes angulares em torno delas.
- Use três hifens em uma linha, '`---`', para indicar um traço longo. TEX os tipografa como um traço longo e os formatadores do Info reduzem três hifens para dois.

Pontos Fora das Aspas

Coloque pontos e outros sinais de pontuação *fora* das citações, a menos que a pontuação faça parte da citação. Essa prática vai contra algumas convenções de publicação nos Estados Unidos, mas habilita o(a) leitor(a) a distinguir entre o conteúdo da citação e a passagem inteira.

Por exemplo, você deveria escrever a seguinte frase com o ponto fora das aspas:

```
Evidentemente, 'au' é uma abreviação para ``autor(a)''.
```

já que '`au`' *não* serve como uma abreviação para '`autor(a)`'. (com um ponto seguinte à palavra).

Apresentando Novos Termos

- Apresente novos termos, de forma que um(a) leitor(a) que não os conheça consiga entendê-los a partir do contexto; ou escreva uma definição para o termo.

  Por exemplo, na seguinte, os termos "check in", "register" e "delta" estão todos aparecendo pela primeira vez; a frase de exemplo deveria ser reescrita, de forma que eles sejam compreensíveis.

  > A função principal te auxilia na submissão inicial de um arquivo no teu sistema de controle de versão e a registrar conjuntos sucessivos de mudanças para ele como deltas.

- Use o comando `@dfn` ao redor de uma palavra sendo apresentada, para indicar que o(a) leitor(a) deveria esperar não saber o significado de antemão, e deveria esperar aprender o significado a partir dessa passagem.

Nós de Invocação de Programa

Você pode invocar programas como Emacs, GCC e `gawk` a partir de um shell. A documentação para cada programa deveria conter uma seção que descreva isso. Infelizmente, se os nomes e títulos dos nós para essas seções forem todos diferentes, eles são difíceis para usuários(as) encontrar.

Portanto, existe uma convenção para nomear essas seções com uma frase que começa com a palavra 'Invocando', como em 'Invocando Emacs'; dessa maneira, os(as) usuários(as) conseguem encontrar a seção facilmente.

Sintaxe ANSI C

Ao usar `@example` para descrever as convenções de chamada de uma função C, use a sintaxe ANSI C, como esta:

```
void dld_init (char *@var{path});
```

E na discussão subsequente, refira-se aos valores dos argumentos escrevendo os mesmos nomes dos argumentos, novamente destacados com `@var`.

Evite o estilo obsoleto que se parece com este:

```
#include <dld.h>

dld_init (path)
  char *path;
```

Além disso, é melhor evitar escrever `#include` acima da declaração apenas para indicar que a função está declarada em um arquivo de cabeçalho. A prática pode dar a impressão equivocada de que o `#include` pertence próximo à declaração da função. Declare explicitamente qual arquivo de cabeçalho contém a declaração ou, melhor ainda, nomeie o arquivo de cabeçalho usado para um grupo de funções no início da seção que descreve as funções.

Comprimento do Nó

Mantenha os nós (seções) em um comprimento razoável, qualquer que seja o razoável no contexto dado. Não hesite em dividir nós longos em sub nós e ter uma extensa estrutura de árvore; é para isso que ela está lá. Muitas vezes, os(as) leitores(as) provavelmente tentarão encontrar um ponto específico no manual, usando busca, indexação ou apenas simples adivinhação, em vez de ler a coisa toda desde o começo até o fim.

Você pode usar o utilitário `texi-elements-by-size` para ver uma lista de todos os nós (ou seções) no documento, ordenados por tamanho (ou por linhas ou por palavras), para encontrar candidatos para divisão. Ele está no subdiretório `util/` dos fontes do Texinfo.

Exemplos Ruins

Aqui estão vários exemplos de escrita ruim para se evitar:

Neste exemplo, digamos, " . . . você precisa `@dfn`{submeter} a nova versão." Isso flui melhor.

> Quando terminar de editar o arquivo, você precisa realizar uma `@dfn`{submissão}.

No exemplo seguinte, digamos, ". . . torna possível uma interface unificada, como o modo VC."

> SCCS, RCS e outros sistemas de controle de versão realizam funções semelhantes de maneiras amplamente semelhantes (é essa semelhança que torna possível um modo de controle unificado como esse).

E neste exemplo, você deveria especificar a que ‘isso’ se refere:

> Se você estiver trabalhando com outras pessoas, isso ajuda a coordenar as mudanças de todas, de forma que elas não atrapalhem umas as outras.

E Finalmente . . .

- Pronuncie TEX como se ‘X’ fosse um ‘chi’ grego, como o último som do nome ‘Bach’. Mas pronuncie Texinfo como em ‘speck’: “teckinfo”.
- Escreva notas para você mesmo(a) no finalzinho de um arquivo do Texinfo depois do `@bye`. Nenhum dos formatadores processa texto depois do `@bye`; é como se o texto estivesse dentro de `@ignore ... @end ignore`.

Apêndice C Arquivos de Amostra do Texinfo

O primeiro exemplo oriundo do primeiro capítulo (veja Seção 2.4 [Amostra Curta], Página 11) é dado aqui na íntegra, sem comentários. O segundo exemplo inclui os textos completos a serem usados nos manuais GNU.

C.1 Amostra Curta

Aqui está um arquivo do Texinfo de amostra completo e curto. Você pode ver esse arquivo, com comentários, no primeiro capítulo. Veja Seção 2.4 [Amostra Curta], Página 11.

Resumindo: O programa `makeinfo` transforma um arquivo fonte do Texinfo como este em um arquivo do Info ou em HTML; e o TEX o tipografa para um manual impresso.

```
\input texinfo
@settitle Manual de Amostra 1.0

@copying
Este é um curto exemplo de um arquivo completo do Texinfo.

Copyright @copyright{} 2016 Free Software Foundation, Inc.
@end copying

@titlepage
@title Título de Amostra
@page
@vskip 0pt plus 1filll
@insertcopying
@end titlepage

@c Exibe o sumário no início.
@contents

@ifnottex
@node Top
@top Amostra GNU

Este manual é para Amostra GNU (version @value{VERSION},
@value{UPDATED}).
@end ifnottex

@menu
* Primeiro Capítulo:: O primeiro capítulo é o único capítulo nesta amostra.
* Índice::            ìndice completo.
@end menu


@node Primeiro Capítulo
@chapter Primeiro Capítulo

@cindex capítulo, primeiro

Este é o primeiro capítulo.
```

```
@cindex entrada de índice, outra

Aqui está uma lista numerada.

@enumerate
@item
Este é o primeiro item.

@item
Este é o segundo item.
@end enumerate


@node Índice
@unnumbered Índice

@printindex cp

@bye
```

C.2 Textos GNU de Amostra

A seguir está uma amostra de documento do Texinfo com os textos completos que deveriam ser usados (adaptados conforme necessário) nos manuais GNU.

Assim como os textos legais, ele também serve como um exemplo prático de quantos elementos em um sistema GNU podem afetar o manual. Se você não está familiarizado(a) com todos esses elementos diferentes, não se preocupe. Eles não são exigidos e um manual perfeitamente bom pode ser escrito sem eles. Eles estão incluídos aqui, no entanto, porque muitos manuais se beneficiam (ou poderiam se beneficiar) deles.

Veja Seção 2.4 [Amostra Curta], Página 11, para um exemplo mínimo de um arquivo do Texinfo. Veja Capítulo 3 [Iniciando e Finalizando um Arquivo], Página 14, para uma explicação completa desse exemplo mínimo.

Aqui estão algumas observações acerca do exemplo:

- O comentário '`$Id:`' é para o CVS (`http://www.nongnu.org/cvs/`), RCS (veja *Revision Control System*) e outros sistemas de controle de versão, que o expandem em uma string como:

  ```
  $Id: texinfo.texi 6987 2016-02-06 08:59:21Z gavin $
  ```

  (Isso é potencialmente útil em todos os fontes que usem controle de versão, não apenas manuais). Você pode desejar incluir o comentário '`$Id:`' no texto do `@copying`, se quiser uma referência completamente inequívoca para a versão do fonte da documentação.

  Se você quiser escrever literalmente `$Id$`, use `@w`: `@w{$}Id$`. Infelizmente, essa técnica não funciona em saída de texto simples, onde não está claro o que deveria ser feito.
- O `version.texi` no comando `@include` é mantido automaticamente pelo Automake (veja *GNU Automake*). Ele configura os valores de '`VERSION`' e de '`UPDATED`' usados em outros lugares. Se tua distribuição não usa o Automake, mas você usa o Emacs, você pode achar o pacote time-stamp.el útil (veja Seção "Time Stamps" em *O Manual do GNU Emacs*).
- O comando `@syncodeindex` reflete a recomendação para usar somente um índice onde possível, para facilitar para os(as) leitores(as) a consulta de entradas de índice.
- O `@dircategory` é para construir o diretório do Info. Veja Seção 21.2.4 [Instalando Entradas de Diretório], Página 190, que inclui uma variedade de nomes recomendados de categorias.

- O nó 'Invocando' é um padrão GNU para ajudar os(as) usuários(as) a encontrar informações básicas relativas ao uso da linha de comando de um programa fornecido. Veja Seção "Manual Structure Details" em *Padrões GNU de Codificação*.
- É melhor incluir a Licença GNU de Documentação Livre inteira em um manual GNU, a menos que o manual tenha somente umas poucas páginas. Claro que esse exemplo é ainda mais curto que isso, mas inclui a FDL de qualquer forma para a finalidade de mostrar uma maneira convencional de fazer isso. O arquivo `fdl.texi` está disponível nas máquinas GNU e nas distribuições de fontes do Texinfo e outras GNU.

  A FDL prevê omissão dela própria sob certas condições, mas nesse caso os textos de amostra fornecidos aqui tem de ser modificados. Veja Apêndice H [Licença GNU de Documentação Livre], Página 265.
- Se a FSF não for a titular dos direitos autorais, então use o nome apropriado.
- Se teu manual for publicado em papel pela FSF ou tiver mais que 400 páginas, você deveria incluir os textos de capa padrão da FSF (veja Seção "License Notices for Documentation" em *Informações de Mantenedor(a) GNU*).
- Para documentos que expressam tuas opiniões, sentimentos ou experiências pessoais, é mais apropriado usar uma licença que permita somente cópias literais, em vez da FDL. Veja Seção C.3 [Licença de Cópia Literal], Página 235.

Aqui está o documento de amostra:

```
\input texinfo   @c -*-texinfo-*-
@comment $Id@w{$}
@comment %**start of header
@include version-pt_BR.texi
@settitle GNU Amostra @value{VERSION}
@syncodeindex pg cp
@comment %**end of header
@copying
Este manual é para GNU Amostra (versão @value{VERSION}, @value{UPDATED}), que é
um exemplo na documentação do Texinfo.

Copyright @copyright{} 2016 Free Software Foundation, Inc.

@quotation
 concedida permissão para copiar, distribuir e (ou) modificar este documentoÉ
sob os termos da Licença GNU de Documentação Livre, Versão 1.3 ou qualquer
versão posterior publicada pela Free Software Foundation; sem Seções
Invariantes, sem Textos de Capa Frontal e sem Textos de Contracapa. Uma cópia
da licença está incluída na seção intitulada ``Licença GNU de Documentação
Livre''.
@end quotation
@end copying

@dircategory Sistema Texinfo de documentação
@direntry
* amostra: (amostra)Invocando amostra.
@end direntry

@titlepage
@title GNU Amostra
@subtitle para versão @value{VERSION}, @value{UPDATED}
```

```
@author A.U. Thor (@email{bug-sample@@gnu.org})
@page
@vskip 0pt plus 1filll
@insertcopying
@end titlepage

@contents

@ifnottex
@node Top
@top GNU Amostra

Este manual é para GNU Amostra (versão @value{VERSION}, @value{UPDATED}).
@end ifnottex

@menu
* Invocando amostra::
* Licença GNU de Documentação Livre::
* Índice::
@end menu


@node Invocando amostra
@chapter Invocando amostra

@pindex amostra
@cindex invocando @command{amostra}

Este é um manual de amostra. Não existe um programa de amostra para invocar,
mas se existisse, você poderia ver o uso básico e opções de linha de comando
dele aqui.


@node Licença GNU de Documentação Livre
@appendix Licença GNU de Documentação Livre

@include fdl-versao_1.3-pt_BR.texi


@node Índice
@unnumbered Índice

@printindex cp

@bye
```

C.3 Licença de Cópia Literal

Para manuais de software e outras documentações, é fundamental usar uma licença que permita redistribuição e atualização livres, de forma que, quando um programa livre for mudado, a documentação também possa ser atualizada.

Por outro lado, para documentos que expressam tuas opiniões, sentimentos ou experiências pessoais, é mais apropriado usar uma licença que permita somente cópias literais.

Aqui está um texto de amostra para tal licença permitindo somente cópias literais. Este é apenas o texto da licença em si. Para um documento completo de amostra, vejam-se as seções anteriores.

```
@copying
Este documento é uma amostra para permitir somente cópias literais.

Copyright @copyright{} 2016 Free Software Foundation, Inc.

@quotation
 concedida permissão para fazer e distribuir cópias literais deste documentoÉ
inteiro sem royalties, desde que o aviso de direitos autorais e este aviso de
permissão sejam preservados.
@end quotation
@end copying
```

C.4 Licença de Cópia Totalmente Permissiva

Para manuais de software e outras documentações, é importante usar uma licença que permita redistribuição e atualização livres, de forma, que quando um programa livre for mudado, a documentação também possa ser atualizada.

Por outro lado, para pequenos arquivos de suporte, manuais curtos (com menos de 300 linhas) e documentação rudimentar (arquivos README, arquivos INSTALL, etc.), a FDL completa seria um exagero. Eles podem usar uma licença simples totalmente permissiva.

Aqui está um texto de amostra para uma tal licença totalmente permissiva. Este é apenas o texto da licença em si. Para um documento completo de amostra, vejam-se as seções anteriores.

```
Copyright @copyright{} 2016 Free Software Foundation, Inc.

A cópia e a distribuição deste arquivo, com ou sem modificação, são permitidas
em qualquer meio, sem royalties, desde que o aviso de direitos autorais e este
aviso sejam preservados.
```

Apêndice D Usando o Modo Texinfo

Você pode editar um arquivo do Texinfo com qualquer editor de texto que escolher. Um arquivo do Texinfo não é diferente de qualquer outro arquivo ASCII. No entanto, o GNU Emacs vem com um modo especial, chamado modo Texinfo, que fornece comandos e ferramentas do Emacs para ajudar a facilitar teu trabalho.

D.1 Visão Geral do Modo Texinfo

O modo Texinfo fornece recursos especiais para trabalhar com arquivos do Texinfo. Você pode:

- Inserir comandos @ usados frequentemente.
- Criar automaticamente linhas `@node`.
- Mostrar a estrutura de um arquivo fonte do Texinfo.
- Criar ou atualizar automaticamente os ponteiros 'Próximo', 'Anterior' e 'Acima' de um nó.
- Criar ou atualizar menus automaticamente.
- Criar automaticamente um menu mestre.
- Formatar uma parte ou todo um arquivo para o Info.
- Tipografar e imprimir parte ou todo um arquivo.

Talvez os dois recursos mais úteis sejam aqueles para inserir comandos @ usados frequentemente e para criar ponteiros de nó e menus.

D.2 Os Comandos Usuais de Edição do GNU Emacs

Na maioria dos casos, os comandos usuais do modo Texto funcionam da mesma forma no modo Texinfo como no modo Texto. O modo Texinfo adiciona novos comandos e ferramentas de edição aos recursos de edição de propósito geral do GNU Emacs. A principal diferença diz respeito ao preenchimento. No modo Texinfo, a variável de separação de parágrafo e a tabela de sintaxe são redefinidas para a finalidade de que os comandos do Texinfo que deveriam estar em linhas próprias não sejam inadvertidamente incluídos em parágrafos. Assim, o comando *M-q* (`fill-paragraph`) preencherá novamente um parágrafo, mas não misturará um comando de indexação em uma linha adjacente a ele no parágrafo.

Além disso, o modo Texinfo configura a variável `page-delimiter` para o valor de `texinfo-chapter-level-regexp`; por padrão, essa é uma expressão regular que corresponde aos comandos para capítulos e os equivalentes deles, como anexos. Com esse valor para o delimitador de página, você pode pular de um título de capítulo para um título de capítulo com os comandos *C-x]* (`forward-page`) e *C-x [* (`backward-page`) e restringir a um capítulo com o comando *C-x n p* (`narrow-to-page`). (Veja Seção "Pages" em *O Manual do GNU Emacs*, para detalhes acerca dos comandos de página).

Você pode nomear um arquivo do Texinfo como desejar, mas a convenção é a de terminar um nome de arquivo do Texinfo com uma das extensões `.texinfo`, `.texi`, `.txi` ou `.tex`. Uma extensão mais longa é preferível, pois é explícita, mas uma extensão mais curta pode ser necessária para sistemas operacionais que limitam o comprimento dos nomes de arquivo. O GNU Emacs automaticamente entra no modo Texinfo quando você visita um arquivo com uma extensão `.texinfo`, `.texi` ou `.txi`. Além disso, o Emacs comuta para o modo Texinfo quando você visita um arquivo que tenha '`-*-texinfo-*-`' na primeira linha dele. Se você já estiver em outro modo e desejar comutar para o modo Texinfo, digite `M-x texinfo-mode`.

Como todos os outros recursos do Emacs, você consegue personalizar ou aprimorar o modo Texinfo como desejar. Em particular, as combinações de teclas são muito fáceis de mudar. As combinações de teclas descritas aqui são as padrões ou as de uso mais comum.

D.3 Inserindo Comandos Usados Frequentemente

O modo Texinfo fornece comandos para inserir vários comandos @ usados frequentemente no buffer. Você pode usar esses comandos para economizar pressionamentos de tecla.

Os comandos de inserção são invocados digitando `C-c` duas vezes e depois a primeira letra do comando @:

`C-c C-c c`
`M-x texinfo-insert-@code`
: Insere `@code{}` e coloca o cursor entre as chaves.

`C-c C-c d`
`M-x texinfo-insert-@dfn`
: Insere `@dfn{}` e coloca o cursor entre as chaves.

`C-c C-c e`
`M-x texinfo-insert-@end`
: Insere `@end` e tenta inserir a palavra correta seguinte, como '`example`' ou '`table`'. (Esse comando não lida com listas aninhadas corretamente, mas insere a palavra apropriada na lista imediatamente precedente).

`C-c C-c i`
`M-x texinfo-insert-@item`
: Insere `@item` e coloca o cursor no início da próxima linha.

`C-c C-c k`
`M-x texinfo-insert-@kbd`
: Insere `@kbd{}` e coloca o cursor entre as chaves.

`C-c C-c n`
`M-x texinfo-insert-@node`
: Insere `@node` e uma linha de comentário listando a sequência para os nós 'Próximo', 'Anterior' e 'Acima'. Deixa um ponto depois do `@node`.

`C-c C-c o`
`M-x texinfo-insert-@noindent`
: Insere `@noindent` e coloca o cursor no início da próxima linha.

`C-c C-c s`
`M-x texinfo-insert-@samp`
: Insere `@samp{}` e coloca o cursor entre as chaves.

`C-c C-c t`
`M-x texinfo-insert-@table`
: Insere `@table` seguido de `SPC` e deixa o cursor depois de `SPC`.

`C-c C-c v`
`M-x texinfo-insert-@var`
: Insere `@var{}` e coloca o cursor entre as chaves.

`C-c C-c x`
`M-x texinfo-insert-@example`
: Insere `@example` e coloca o cursor no início da próxima linha.

`C-c C-c {`
`M-x texinfo-insert-braces`
: Insere `{}` e coloca o cursor entre as chaves.

C-c }
C-c]
M-x up-list
: Move de entre um par de chaves para a frente, passando pela chave de fechamento. Digitar *C-c]* é mais fácil que digitar *C-c }*, que é, no entanto, mais mnemônico; daí as duas combinações de teclas. (Além disso, você pode sair de entre chaves digitando *C-f*).

Para colocar um comando como `@code{...}` em volta de uma palavra *existente*, posicione o cursor na frente da palavra e digite *C-u 1 C-c C-c c*. Isso facilita editar texto simples existente. O valor do argumento de prefixo informa ao Emacs quantas palavras seguintes ao ponto incluir entre chaves—'1' para uma palavra, '2' para duas palavras e assim por diante. Use um argumento negativo para cercar a palavra ou palavras anterior. Se você não especificar um argumento de prefixo, o Emacs insere a string do comando @ e posiciona o cursor entre as chaves. Esse recurso funciona somente para os comandos @ que operam sobre uma palavra ou palavras dentro de uma linha, como `@kbd` e `@var`.

Esse conjunto de comandos de inserção foi criado depois de analisar a frequência com que diferentes comandos @ são usados no *Manual do GNU Emacs* e no *Manual do GDB*. Se desejar adicionar teus próprios comandos de inserção, você pode vincular uma macro de teclado a uma tecla, usar abreviações ou estender o código em `texinfo.el`.

C-c C-c C-d (`texinfo-start-menu-description`) é um comando de inserção que funciona diferentemente dos outros comandos de inserção. Ele insere o título da seção ou o do capítulo de um nó no espaço para a descrição em uma linha de entrada de menu. (Uma entrada de menu tem três partes, o nome da entrada, o nome do nó e a descrição. Somente o nome do nó é exigido, mas uma descrição ajuda a explicar do que se trata o nó. Veja Seção 4.9.4 [Partes de Menu], Página 37).

Para usar `texinfo-start-menu-description`, posicione o ponto em uma linha de entrada de menu e digite *C-c C-c C-d*. O comando procura e copia o título que acompanha o nome do nó e insere o título como uma descrição; ele posiciona o ponto no início do texto inserido para que você possa editá-lo. A função não insere o título se a linha de entrada do menu já contiver uma descrição.

Esse comando é somente um auxílio para escrever descrições; ele não faz o trabalho todo. Você precisa editar o texto inserido, pois um título tende a usar as mesmas palavras que um nome de nó, mas uma descrição útil usa palavras diferentes.

D.4 Mostrando a Estrutura de Seccionamento de um Arquivo

Você pode mostrar a estrutura de seccionamento de um arquivo Texinfo usando o comando *C-c C-s* (`texinfo-show-structure`). Este comando lista as linhas que começam com os comandos @ para `@chapter`, `@section` e similares. Ele constrói o que equivale a um índice. Essas linhas são exibidas em outro buffer chamado buffer '`*Occur*`'. Nesse buffer, você pode posicionar o cursor sobre uma das linhas e usar o comando *C-c C-c* (`occur-mode-goto-occurrence`) para pular para o ponto correspondente no arquivo Texinfo.

C-c C-s
M-x texinfo-show-structure
: Mostra o `@chapter`, `@section` e linhas semelhantes de um arquivo do Texinfo.

C-c C-c
M-x occur-mode-goto-occurrence
: Vai para a linha no arquivo do Texinfo correspondente à linha sob o cursor no buffer `*Occur*`.

Se você chamar `texinfo-show-structure` com um argumento de prefixo digitando *C-u C-c C-s*, ele listará não somente aquelas linhas com os comandos @ para `@chapter`, `@section` e similares, mas também as linhas `@node`. Você pode usar `texinfo-show-structure` com um argumento de prefixo para verificar se os ponteiros 'Próximo', 'Anterior' e 'Acima' de uma linha `@node` estão corretos.

Frequentemente, quando você está trabalhando em um manual, você estará interessado(a) somente na estrutura do capítulo atual. Nesse caso, você pode marcar a região do buffer na qual você está interessado(a) usando o comando *C-x n n* (`narrow-to-region`) e `texinfo-show-structure` funcionará somente naquela região. Para ver o buffer inteiro novamente, use *C-x n w* (`widen`). (Veja Seção "Narrowing" em *O Manual do GNU Emacs*, para mais informações acerca dos comandos de estreitamento).

Além de fornecer o comando `texinfo-show-structure`, o modo Texinfo configura o valor da variável delimitadora de página para corresponder aos comandos @ de nível de capítulo. Isso te habilita a usar os comandos *C-x]* (`forward-page`) e *C-x [* (`backward-page`) para avançar e retroceder por capítulo, e a usar o comando *C-x n p* (`narrow-to-page`) para restringir a um capítulo. Veja Seção "Pages" em *O Manual do GNU Emacs*, para mais informações acerca dos comandos de página.

D.5 Atualizando Nós e Menus

O modo Texinfo fornece comandos para criar ou atualizar automaticamente menus e ponteiros de nó. Os comandos são chamados de comandos de "atualizar" porque o uso mais frequente deles é para atualizar um arquivo do Texinfo depois que você trabalhou nele; mas você pode usá-los para inserir os ponteiros "Próximo', 'Anterior' e 'Acima' em uma linha `@node` que não tenha nenhum e para criar menus em um arquivo que não tenha nenhum.

Se não usar quaisquer comandos de atualização, você precisará escrever menus manualmente, o que é uma tarefa tediosa.

D.5.1 Os Comandos de Atualização

Você pode usar os comandos de atualização para:

- inserir ou atualizar os ponteiros 'Próximo', 'Anterior' e 'Acima' de um nó,
- inserir ou atualizar o menu de uma seção e
- criar um menu mestre para um arquivo fonte do Texinfo.

Você também pode usar os comandos para atualizar todos os nós e menus em uma região ou em um arquivo do Texinfo inteiro.

Os comandos de atualização funcionam somente com arquivos do Texinfo convencionais, que são estruturados hierarquicamente como livros. Em tais arquivos, uma linha de comando de estruturação precisa seguir próximo depois de cada linha `@node`, exceto para a linha 'Top' do `@node`. (Uma *linha de comando de estruturação* é uma linha que começa com `@chapter`, `@section` ou outro comando similar).

Você pode escrever a linha de comando de estruturação na linha que segue imediatamente depois uma linha `@node` ou então na linha que segue depois de uma linha `@comment` ou uma linha `@ifinfo`. Você não pode interpor mais que uma linha entre a linha `@node` e a linha de comando de estruturação; e você pode interpor somente uma linha `@comment` ou uma linha `@ifinfo`.

Comandos que funcionam em um buffer inteiro exigem que o nó 'Top' seja seguido por um nó com um comando `@chapter` ou nível equivalente. Os comandos de atualização de menu não criarão um menu principal ou mestre para um arquivo do Texinfo que tenha somente nós de nível `@chapter`! Os comandos de atualização de menu criam somente menus *dentro de* nós para nós de nível inferior. Para criar um menu de capítulos, você precisa fornecer um nó 'Top'.

Os comandos de atualização de menu removem entradas de menu que se referem a outros arquivos do Info, pois eles não se referem a nós dentro do buffer atual. Isso é uma deficiência. Em vez de usar entradas de menu, você pode usar referências cruzadas para se referir a outros arquivos do Info. Nenhum dos comandos de atualização afeta referências cruzadas.

O modo Texinfo tem cinco comandos de atualização que são usados com mais frequência: dois são para atualizar os ponteiros de nó ou o menu de um nó (ou de uma região); dois são para atualizar cada ponteiro de nó e menu em um arquivo; e um, o comando `texinfo-master-menu`, é para criar um menu mestre para um arquivo completo e, opcionalmente, para atualizar cada nó e menu no arquivo do Texinfo inteiro.

O comando `texinfo-master-menu` é o comando principal:

C-c C-u m
M-x texinfo-master-menu

Cria ou atualiza um menu mestre que inclua todos os outros menus (incorporando as descrições oriundas dos menus preexistentes, se existir).

Com um argumento (argumento de prefixo, *C-u,* se interativo), primeiro crie ou atualize todos os nós e todos os menus regulares no buffer antes de construir o menu mestre. (Veja Seção 3.6 [O Nó Top e o Menu Mestre], Página 23, para mais acerca de um menu mestre).

Para `texinfo-master-menu` funcionar, o arquivo do Texinfo precisa ter um nó 'Top' e pelo menos um nó subsequente.

Depois de editar extensivamente um arquivo do Texinfo, você pode digitar o seguinte:

```
C-u M-x texinfo-master-menu
```

ou

```
C-u C-c C-u m
```

Isso atualiza todos os nós e menus completamente e de uma vez.

Os outros principais comandos de atualização realizam trabalhos menores e são projetados para a pessoa que atualiza nós e menus conforme ele ou ela escreve um arquivo do Texinfo.

Os comandos são:

C-c C-u C-n
M-x texinfo-update-node

Insere os ponteiros 'Próximo', 'Anterior' e 'Acima' para o nó em que o ponto está (ou seja, para a linha `@node` precedente ao ponto). Se a linha `@node` tiver ponteiros 'Próximo', 'Anterior' ou 'Acima' preexistentes, os ponteiros antigos serão removidos e novos serão inseridos. Com um argumento (argumento de prefixo, *C-u*, se interativo), esse comando atualiza todas as linhas `@node` na região (que é o texto entre o ponto e a marca).

C-c C-u C-m
M-x texinfo-make-menu

Cria ou atualiza o menu no nó em que o ponto está. Com um argumento (*C-u* como argumento de prefixo, se interativo), o comando cria ou atualiza menus para os nós que estão dentro ou são parte da região.

Sempre que `texinfo-make-menu` atualiza um menu existente, as descrições oriundas desse menu são incorporadas ao novo menu. Isso é feito copiando descrições oriundas do menu existente para as entradas no novo menu que tenham os mesmos nomes de nó. Se os nomes de nó forem diferentes, as descrições não são copiadas para o novo menu.

C-c C-u C-e
M-x texinfo-every-node-update
: Insere ou atualiza os ponteiros 'Próximo', 'Anterior' e 'Acima' para cada nó no buffer.

C-c C-u C-a
M-x texinfo-all-menus-update
: Cria ou atualiza todos os menus no buffer. Com um argumento (*C-u* como argumento de prefixo, se interativo), primeiro insere ou atualiza todos os ponteiros de nó antes de trabalhar nos menus.

  Se um menu mestre existir, o comando `texinfo-all-menus-update` o atualiza; mas o comando não cria um novo menu mestre se nenhum existir. (Use o comando `texinfo-master-menu` para isso).

  Ao trabalhar em um documento que não merece um menu mestre, você pode digitar o seguinte:

  ```
  C-u C-c C-u C-a
  ```

  or

  ```
  C-u M-x texinfo-all-menus-update
  ```

  Isso atualiza todos os nós e menus.

A variável `texinfo-column-for-description` especifica a coluna para a qual as descrições do menu são recuadas. Por padrão, o valor é 32, embora possa ser útil reduzi-lo para até 24. Você pode configurar a variável por meio de personalização (veja Seção "Customization" em *O Manual do GNU Emacs*) ou com o comando *M-x set-variable* (veja Seção "Examining and Setting Variables" em *O Manual do GNU Emacs*).

Além disso, o comando `texinfo-indent-menu-description` pode ser usado para recuar descrições de menu existentes para uma coluna especificada. Finalmente, se desejar, você pode usar o comando `texinfo-insert-node-lines` para inserir linhas `@node` ausentes em um arquivo. (Veja Seção D.5.3 [Outros Comandos de Atualização], Página 243, para mais informações).

D.5.2 Exigências de Atualização

Para usar os comandos de atualização, você precisa organizar o arquivo do Texinfo hierarquicamente com capítulos, seções, subseções e similares. Quando você construir a hierarquia do manual, não 'pule para baixo' mais que um nível por vez: você pode seguir o nó 'Top' com um capítulo, mas não com uma seção; você pode seguir um capítulo com uma seção, mas não com uma subseção. No entanto, você pode 'pular para cima' qualquer número de níveis ao mesmo tempo—por exemplo, desde uma subseção para um capítulo.

Cada linha `@node`, com exceção da linha para o nó 'Top', precisa ser seguida por uma linha com um comando de estruturação, como `@chapter`, `@section` ou `@unnumberedsubsec`.

Cada combinação de linha `@node` e de comando de estruturação precisa ser semelhante a esta:

```
@node     Comentários,  Mínima, Convenções, Visão Geral
@comment  nome-nó, próximo,    anterior,    acima
@section Comentários
```

ou a esta (sem a linha `@comment`):

```
@node Comentários, Mínima, Convenções, Visão Geral
@section Comentários
```

ou a esta (sem os ponteiros explícitos de nó):

```
@node Comentários
@section Comentários
```

Nesse exemplo, 'Comentários' é o nome do nó e o da seção. O próximo nó é chamado 'Mínima' e o nó anterior é chamado 'Convenções'. A seção 'Comentários' está dentro do nó 'Visão Geral',

que é especificado pelo ponteiro 'Acima'. (Em vez de uma linha `@comment`, você também pode escrever uma linha `@ifinfo`).

Se um arquivo tiver um nó 'Top', ele precisa ser chamado de '`top`' ou de '`Top`' e ser o primeiro nó no arquivo.

Os comandos de atualização de menu criam um menu de seções dentro de um capítulo, um menu de subseções dentro de uma seção, e assim por diante. Isso significa que você precisa ter um nó 'Top' se quiser um menu de capítulos.

A propósito, o comando `makeinfo` criará um arquivo do Info para um arquivo do Texinfo organizado hierarquicamente que careça de ponteiros 'Próximo', 'Anterior' e 'Acima'. Assim, se puder ter certeza de que teu arquivo do Texinfo será formatado com `makeinfo`, você não tem necessidade dos comandos de atualização de nó. (Veja Seção 21.1 [Criando um Arquivo do Info], Página 185, para mais informações acerca do `makeinfo`).

D.5.3 Outros Comandos de Atualização

Além dos cinco principais comandos de atualização, o modo Texinfo possui vários comandos de atualização usados menos frequentemente:

M-x texinfo-insert-node-lines
: Insere as linhas `@node` antes de `@chapter`, `@section` e de outros comandos de seccionamento sempre que estiverem ausentes em uma região em um arquivo do Texinfo.

  Com um argumento (*C-u* como argumento de prefixo, se interativo), o comando `texinfo-insert-node-lines` não somente insere linhas `@node`, mas também insere os títulos de capítulos ou de seções como os nomes dos nós correspondentes. Além disso, ele insere os títulos como nomes de nós em linhas `@node` preexistentes que careçam de nomes. Como os nomes de nós deveriam ser mais concisos que os títulos de seções ou de capítulos, você precisa editar manualmente os nomes de nós assim inseridos.

  Por exemplo, o seguinte marca um buffer inteiro como uma região e insere linhas e títulos `@node` por toda parte:

  ```
  C-x h C-u M-x texinfo-insert-node-lines
  ```

  Esse comando insere títulos como nomes de nós em linhas `@node`; o comando `texinfo-start-menu-description` (veja Seção D.3 [Inserindo], Página 238) insere títulos como descrições em entradas de menu, uma ação diferente. No entanto, em ambos os casos, você precisa editar o texto inserido.

M-x texinfo-multiple-files-update
: Atualiza nós e menus em um documento construído a partir de arquivos separados. Com *C-u* como um argumento de prefixo, cria e insere um menu mestre no arquivo externo. Com um argumento de prefixo numérico, como *C-u 2*, primeiro atualiza todos os menus e todos os ponteiros 'Próximo', 'Anterior' e 'Acima' de todos os arquivos incluídos antes de criar e inserir um menu mestre no arquivo externo. O comando `texinfo-multiple-files-update` está descrito no anexo acerca de arquivos `@include`. Veja Seção 18.2 [`texinfo-multiple-files-update`], Página 146.

M-x texinfo-indent-menu-description
: Recua cada descrição no menu seguindo o ponto para a coluna especificada. Você pode usar esse comando para ter mais espaço para descrições. Com um argumento (*C-u* como argumento de prefixo, se interativo), o comando `texinfo-indent-menu-description` recua cada descrição em cada menu na região. No entanto, esse comando não recua a segunda linha e as subsequentes de uma descrição de várias linhas.

M-x texinfo-sequential-node-update
: Insere os nomes dos nós imediatamente após e antes do nó atual como os ponteiros 'Próximo' ou 'Anterior', independentemente do nível hierárquico desses nós. Isso significa que o nó 'Próximo' de uma subseção pode muito bem ser o próximo capítulo. Nós sequencialmente ordenados são úteis para romances e outros documentos que você lê sequencialmente. (No entanto, no Info, o comando *g* * permite que você olhe ao longo do arquivo sequencialmente, de forma que nós sequencialmente ordenados não são estritamente necessários). Com um argumento (argumento de prefixo, se interativo), o comando `texinfo-sequential-node-update` atualiza sequencialmente todos os nós na região.

D.6 Formatação para Info

O modo Texinfo fornece vários comandos para formatar parte ou todo um arquivo do Texinfo para Info. Frequentemente, quando estiver escrevendo um documento, você vai querer formatar somente parte de um arquivo—ou seja, uma região.

Você pode usar o comando `texinfo-format-region` ou `makeinfo-region` para formatar uma região:

C-c C-e C-r
M-x texinfo-format-region
C-c C-m C-r
M-x makeinfo-region
: Formata a região atual para Info.

Você pode usar ou o comando `texinfo-format-buffer` ou o `makeinfo-buffer` para formatar um buffer inteiro:

C-c C-e C-b
M-x texinfo-format-buffer
C-c C-m C-b
M-x makeinfo-buffer
: Formata o buffer atual para Info.

Por exemplo, depois de escrever um arquivo do Texinfo, você pode digitar o seguinte:

```
C-u C-c C-u m
```

or

```
C-u M-x texinfo-master-menu
```

Isso atualiza todos os nós e menus. Então digitar o seguinte para criar um arquivo do Info:

```
C-c C-m C-b
```

ou

```
M-x makeinfo-buffer
```

Veja Seção 21.1 [Criando um Arquivo do Info], Página 185, para detalhes acerca da formatação Info.

D.7 Impressão

Tipografar e imprimir um arquivo do Texinfo é um processo de várias etapas no qual você primeiro cria um arquivo para impressão (chamado de arquivo DVI) e então imprime o arquivo. Opcionalmente, você também pode criar índices. Para fazer isso, você precisa executar o comando `texindex` depois de executar primeiro o comando de tipografia `tex`; e então você precisa executar o comando `tex` novamente. Ou então execute o comando `texi2dvi` que cria índices automaticamente conforme necessário (veja Seção 19.2 [Formatar com `texi2dvi`], Página 150).

Frequentemente, quando você está escrevendo um documento, você quer tipografar e imprimir somente parte de um arquivo para ver como ele ficará. Você pode usar o `texinfo-tex-region` e comandos relacionados para esse propósito. Use o comando `texinfo-tex-buffer` para formatar todo um buffer.

C-c C-t C-b
M-x texinfo-tex-buffer
: Executa `texi2dvi` no buffer. Além de executar TEX no buffer, esse comando cria ou atualiza índices automaticamente conforme necessário.

C-c C-t C-r
M-x texinfo-tex-region
: Executa TEX a região.

C-c C-t C-i
M-x texinfo-texindex
: Executa `texindex` para ordenar os índices de um arquivo do Texinfo formatado com `texinfo-tex-region`. O comando `texinfo-tex-region` não executa `texindex` automaticamente; ele executa somente o comando de tipografia `tex`. Você precisa executar o comando `texinfo-tex-region` uma segunda vez depois de ordenar os arquivos de índices brutos com o comando `texindex`. (Normalmente, você não formata um índice quando formata uma região, somente quando formata um buffer. Agora que o comando `texi2dvi` existe, existe pouca ou nenhuma necessidade para esse comando).

C-c C-t C-p
M-x texinfo-tex-print
: Imprime o arquivo (ou parte do arquivo) formatado anteriormente com `texinfo-tex-buffer` ou `texinfo-tex-region`.

Para `texinfo-tex-region` ou `texinfo-tex-buffer` funcionarem, o arquivo *precisa* começar com uma linha '`\input texinfo`' e precisa incluir uma linha `@settitle`. O arquivo precisa terminar com `@bye` em uma linha própria. (Quando você usa `texinfo-tex-region`, você precisa cercar a linha `@settitle` com linhas start-of-header e end-of-header).

Veja Capítulo 19 [Impresso], Página 150, para uma descrição dos outros comandos relacionados ao TEX, como `tex-show-print-queue`.

D.8 Resumo do Modo Texinfo

No modo Texinfo, cada conjunto de comandos tem atalhos de teclado padrão que começam com as mesmas teclas. Todos os comandos que são criados sob medida para o modo Texinfo começam com *C-c*. As teclas são um tanto mnemônicas.

Comandos de Inserção

Os comandos de inserção são invocados digitando *C-c* duas vezes e, em seguida, a primeira letra do comando @ a ser inserido. (Pode fazer mais sentido mnemonicamente usar *C-c C-i*, para 'inserção personalizada', mas *C-c C-c* é rápido de digitar).

`C-c C-c c`	Inserir '`@code`'.
`C-c C-c d`	Inserir '`@dfn`'.
`C-c C-c e`	Inserir '`@end`'.
`C-c C-c i`	Inserir '`@item`'.
`C-c C-c n`	Inserir '`@node`'.
`C-c C-c s`	Inserir '`@samp`'.
`C-c C-c v`	Inserir '`@var`'.

`C-c {` Inserir chaves.
`C-c ]`
`C-c }` Sair das chaves de cercamento.

`C-c C-c C-d` Inserir título da seção de um nó no espaço para a descrição em uma linha de entrada de menu.

Mostrar Estrutura

O comando `texinfo-show-structure` é frequentemente usado dentro de uma região restrita.

`C-c C-s` Listar todos os títulos.

O Comando Mestre de Atualização

O comando `texinfo-master-menu` cria um menu mestre e também pode ser usado para atualizar cada nó e menu em um arquivo.

`C-c C-u m`
`M-x texinfo-master-menu`
Cria ou atualiza um menu mestre.

`C-u C-c C-u m` Com *`C-u`* como argumento de prefixo, primeiro cria ou atualiza todos os nós e menus regulares, e então cria um menu mestre.

Atualizar Ponteiros

Os comandos de atualização de ponteiros são invocados digitando *`C-c C-u`* e então *`C-n`* para `texinfo-update-node` ou *`C-e`* para `texinfo-every-node-update`.

`C-c C-u C-n` Atualiza um nó.
`C-c C-u C-e` Atualiza cada nó no buffer.

Atualizar Menus

Invoque os comandos de atualização de menus digitando *`C-c C-u`* e então *`C-m`* para `texinfo-make-menu` ou *`C-a`* para `texinfo-all-menus-update`. Para atualizar nós e menus ao mesmo tempo, preceda *`C-c C-u C-a`* com *`C-u`*.

`C-c C-u C-m` Faz ou atualiza um menu.

`C-c C-u C-a` Faz ou atualiza todos menus em um buffer.

`C-u C-c C-u C-a` Com *`C-u`* como argumento de prefixo, primeiro cria ou atualiza todos os nós e então cria ou atualiza todos os menus.

Formatar para Info

Os comandos de formatação Info que estão escritos em Emacs Lisp são invocados digitando *`C-c C-e`* e então *`C-r`* para uma região ou *`C-b`* para o buffer inteiro.

Os comandos de formatação Info que estão escritos em C e baseados no programa `makeinfo` são invocados digitando *`C-c C-m`* e então *`C-r`* para uma região ou *`C-b`* para o buffer inteiro.

Use os comandos `texinfo-format...`:

`C-c C-e C-r` Formatar a região.
`C-c C-e C-b` Formatar o buffer.

Use `makeinfo`:

`C-c C-m C-r` Formatar a região.
`C-c C-m C-b` Formatar o buffer.
`C-c C-m C-l` Recentralizar o buffer de saída do `makeinfo`.
`C-c C-m C-k` Mata o trabalho de formatação `makeinfo`.

Tipografia e Impressão

Os comandos de tipografia e impressão do TEX são invocados digitando *`C-c C-t`* e depois outro comando de controle: *`C-r`* para `texinfo-tex-region`, *`C-b`* para `texinfo-tex-buffer` e assim por diante.

`C-c C-t C-r` Executa TEX na região.
`C-c C-t C-b` Executa `texi2dvi` no buffer.
`C-c C-t C-i` Executa `texindex.`
`C-c C-t C-p` Imprime o arquivo DVI.
`C-c C-t C-q` Mostra a fila de impressão.
`C-c C-t C-d` Deleta um trabalho da fila de impressão.
`C-c C-t C-k` Mata o trabalho atual de formatação do TEX.
`C-c C-t C-x` Sai de um trabalho de formatação TEX atualmente parado.
`C-c C-t C-l` Recentraliza o buffer de saída.

Outros Comandos de Atualização

Os comandos de atualização restantes não tem atalhos padrão de teclado porque raramente são usados.

`M-x texinfo-insert-node-lines`
Insere linhas `@node` ausentes na região.
Com *`C-u`* como um argumento de prefixo,
usa títulos de seção como nomes de nó.

`M-x texinfo-multiple-files-update`
Atualizar um documento multi arquivos.
Com *`C-u 2`* como argumento de prefixo,
cria ou atualiza todos os nós e menus
em todos os arquivos incluídos primeiro.

`M-x texinfo-indent-menu-description`
Recua descrições.

`M-x texinfo-sequential-node-update`
Insere ponteiros de nó em sequência estrita.

Apêndice E Cabeçalhos de Página

A maioria dos manuais impressos contém títulos no topo de cada página, exceto as páginas de título e copyright. Alguns manuais também contém rodapés.

E.1 Cabeçalhos Introduzidos

O Texinfo fornece formatos padrão de cabeçalho de página para manuais que sejam impressos em um lado de cada folha de papel e para manuais que sejam impressos em ambos os lados do papel. Tipicamente, você usará esses formatos, mas pode especificar teu próprio formato se desejar.

Além disso, você pode especificar se os capítulos deveriam começar em uma nova página ou simplesmente continuar na mesma página que o capítulo anterior; e se os capítulos começarem em novas páginas, você pode especificar se eles precisam ser páginas ímpares.

Por convenção, um livro é impresso em ambos os lados de cada folha de papel. Quando você abre um livro, a página da direita é numerada de forma ímpar, e os capítulos começam nas páginas da direita—uma página anterior da esquerda é deixada em branco, se necessário. Informes, no entanto, frequentemente são impressos em apenas um lado do papel, e os capítulos começam em uma nova página imediatamente seguinte ao fim do capítulo anterior. Em informes curtos ou informais, os capítulos frequentemente não começam em uma nova página, mas são separados do texto precedente por uma pequena quantidade de espaço em branco.

O comando `@setchapternewpage` controla se os capítulos começam em novas páginas e se um dos formatos padrão de título é usado. Além disso, o Texinfo tem vários comandos de título e rodapé que você pode usar para gerar teus próprios formatos de título e rodapé.

No Texinfo, títulos e rodapés são linhas unitárias nas partes superior e inferior das páginas; você não pode criar títulos ou rodapés multilinha. Cada linha de cabeçalho ou rodapé é dividida em três partes: uma parte esquerda, uma parte do meio e uma parte direita. Qualquer parte, ou uma linha inteira, pode ser deixada em branco. O texto para a parte esquerda de uma linha de cabeçalho ou rodapé é configurado alinhado à esquerda; o texto para a parte do meio é centralizado; e o texto para a parte direita é configurado alinhado à direita.

E.2 Formatos de Título de Uso Comum

O Texinfo fornece dois formatos de cabeçalho de uso mais comum, um para manuais impressos em um lado de cada folha de papel e outro para manuais impressos em ambos os lados do papel.

Por padrão, nada é especificado para o rodapé de um arquivo do Texinfo, de forma que o rodapé permanece em branco.

O formato de uso mais comum para impressão em um lado consiste de uma linha de cabeçalho na qual a parte esquerda contém o nome do capítulo, a parte central fica em branco e a parte direita contém o número da página.

Uma página de um lado se parece com isto:

```
 -------------------------
|                         |
| capítulo  número página |
|                         |
| Início do texto ...     |
| ...                     |
|                         |
```

O formato de uso mais comum para impressão frente e verso depende se o número da página é par ou ímpar. Por convenção, páginas pares ficam à esquerda e páginas ímpares ficam à

direita. (TEX ajustará as larguras das margens esquerda e direita. Normalmente, as larguras estão corretas, mas durante a impressão frente e verso, é sensato verificar se as páginas serão encadernadas corretamente—ocasionalmente, uma impressora produzirá uma saída na qual as páginas pares tem uma margem direita maior que as páginas ímpares).

No formato de uso mais comum frente e verso, a parte esquerda da página esquerda (numeração par) contém o número da página, a parte central está em branco e a parte direita contém o título (especificado pelo comando `@settitle`). A parte esquerda da página direita (numeração ímpar) contém o nome do capítulo, a parte central fica em branco e a parte direita contém o número da página.

Duas páginas, lado a lado, como em um livro aberto, se parecem com isto:

```
 ----------------------     -------------------------
|                      |   |                         |
| número página  título |   | capítulo número página |
|                      |   |                         |
| Início do texto ...  |   | Mais  texto ...         |
| ...                  |   | ...                     |
|                      |   |                         |
```

O nome do capítulo é precedido pela palavra "Capítulo", o número do capítulo e dois pontos. Isso torna mais fácil acompanhar onde você está no manual.

E.3 Especificando o Tipo de Título

TEX não começa a gerar títulos de página para um arquivo do Texinfo de uso mais comum até que ele alcance o comando `@end titlepage`. Portanto, as páginas de título e copyright não são numeradas. O comando `@end titlepage` faz com que o TEX comece a gerar títulos de página de acordo com um formato de uso mais comum especificado pelo comando `@setchapternewpage` que precede a seção `@titlepage`.

Existem quatro possibilidades:

Sem comando `@setchapternewpage`
: Faz com que TEX especifique o formato de título de um lado, com capítulos em novas páginas. Isso é o mesmo que `@setchapternewpage on`.

`@setchapternewpage on`
: Especifica o formato de título de lado unitário, com capítulos em novas páginas.

`@setchapternewpage off`
: Faz com que o TEX inicie um novo capítulo na mesma página da última página do capítulo precedente, depois de pular alguns espaços verticais em branco. Também faz com que o TEX tipografe para impressão de um lado unitário. (Você pode substituir o formato dos cabeçalhos com o comando `@headings double`; veja Seção 3.7.3 [`@headings`], Página 26).

`@setchapternewpage odd`
: Especifica o formato de título frente e verso, com capítulos em novas páginas.

Texinfo carece de um comando `@setchapternewpage even`.

E.4 Como Fazer Teus Próprios Cabeçalhos

Você pode usar os títulos de uso mais comum fornecidos com o Texinfo ou especificar os teus próprios. Por padrão, o Texinfo não tem rodapés, de forma que se você especificá-los, o tamanho de página disponível para o texto principal será ligeiramente reduzido.

O Texinfo fornece seis comandos para especificar títulos e rodapés:

- `@everyheading` e `@everyfooting` geram cabeçalhos e rodapés de página que são os mesmos para páginas pares e ímpares.
- Os comandos `@evenheading` e `@evenfooting` geram cabeçalhos e rodapés para páginas pares (lado esquerdo).
- `@oddheading` e `@oddfooting` geram cabeçalhos e rodapés para páginas ímpares (lado direito).

Escreva especificações personalizadas de título no arquivo do Texinfo imediatamente depois do comando `@end titlepage`. Você precisa cancelar os comandos de título predefinidos com o comando `@headings off` antes de definir tuas próprias especificações.

Aqui está como dizer ao TEX para colocar o nome do capítulo à esquerda, o número da página no centro e a data à direita de cada cabeçalho para páginas pares e ímpares:

```
@headings off
@everyheading @thischapter @| @thispage @| @today{}
```

Você precisa dividir a parte esquerda da parte central e a parte central da parte direita inserindo '`@|`' entre as partes. Caso contrário, o comando de especificação não será capaz de dizer onde o texto de uma parte termina e a próxima parte começa.

Cada parte pode conter texto ou comandos @. O texto é impresso como se a parte estivesse dentro de um parágrafo comum no corpo da página. Os comandos @ substituem eles próprios pelo número da página, data, nome do capítulo ou o que for.

Aqui estão os seis comandos de título e rodapé:

`@everyheading` *left* `@|` *center* `@|` *right*
`@everyfooting` *left* `@|` *center* `@|` *right*
: Os comandos 'every' especificam o formato para páginas pares e ímpares. Esses comandos são para documentos que sejam impressos em um lado de cada folha de papel, ou para documentos nos quais você queira cabeçalhos ou rodapés simétricos.

`@evenheading` *left* `@|` *center* `@|` *right*
`@oddheading` *left* `@|` *center* `@|` *right*
`@evenfooting` *left* `@|` *center* `@|` *right*
`@oddfooting` *left* `@|` *center* `@|` *right*
: Os comandos 'even' e 'odd' especificam o formato para páginas pares e páginas ímpares. Esses comandos são para livros e manuais que sejam impressos em ambos os lados de cada folha de papel.

Use a série '`@this...`' de comandos @ para fornecer os nomes de capítulos e seções e o número da página. Você pode usar os comandos '`@this...`' nas partes esquerda, central ou direita dos cabeçalhos e rodapés, ou em qualquer outro lugar em um arquivo do Texinfo, desde que estejam entre os comandos `@iftex` e `@end iftex`.

Aqui estão os comandos '`@this...`':

`@thispage`
: Expande para o número da página atual.

`@thissectionname`
: Expande para o nome da seção atual.

`@thissectionnum`
: Expande para o número da seção atual.

`@thissection`
: Expande para o número e nome da seção atual, no formato 'Seção 1: Título'.

`@thischaptername`
: Expande para o nome do capítulo atual.

`@thischapternum`
: Expande para o número do capítulo atual ou letra do anexo atual.

`@thischapter`
: Expande para o número e nome do capítulo atual, no formato 'Capítulo 1: Título'.

`@thistitle`
: Expande para o nome do documento, conforme especificado pelo comando `@settitle`.

`@thisfile`
: Somente para arquivos `@include`: expande para o nome do arquivo `@include` atual. Se o arquivo fonte atual do Texinfo não for um arquivo `@include`, esse comando não terá efeito. Esse comando *não* fornece o nome do arquivo fonte atual do Texinfo, a menos que seja um arquivo `@include`. (Veja Capítulo 18 [Arquivos de Inclusão], Página 146, para mais informações acerca de arquivos `@include`).

Você também pode usar o comando `@today{}`, que expande para a data atual, no formato '1 Jan 1900'.

Outros comandos @ e texto são impressos em um cabeçalho ou rodapé exatamente como se estivessem no corpo de uma página. É útil para incorporar texto, particularmente quando você estiver escrevendo rascunhos:

```
@headings off
@everyheading @emph{Rascunho!} @| @thispage @| @thischapter
@everyfooting @| @| Versão: 0.27: @today{}
```

Cuidado com títulos muito longos: eles podem se sobrepor a outra parte do cabeçalho ou rodapé e apagá-la.

Se você tiver capítulos e (ou) seções muito curtos, vários deles podem aparecer em uma página. Você pode especificar a quais capítulos e seções você quer que `@thischapter`, `@thissection` e outras macros se refiram em tais páginas, como segue:

`@everyheadingmarks` *referência*
`@everyfootingmarks` *referência*
: O argumento *referência* pode ser `top` (os comandos `@this...` se referirão ao capítulo/seção no topo de uma página) ou `bottom` (os comandos refletirão a situação no final de uma página). Esses comandos '`@every...`' especificam o que fazer em páginas pares e ímpares.

`@evenheadingmarks` *referência*
`@oddheadingmarks` *referência*
`@evenfootingmarks` *referência*
`@oddfootingmarks` *referência*
: Esses comandos '`@even...`' e '`@odd...`' especificam o que fazer somente em páginas pares ou ímpares, respectivamente. O argumento *referência* é o mesmo dos comandos '`@every...`'.

Escreva esses comandos imediatamente depois dos comandos `@...contents` ou depois do comando `@end titlepage` se você não tiver um sumário ou se ele estiver impresso no final do teu manual.

Por padrão, os comandos `@this...` refletem a situação na parte inferior de uma página, tanto nos títulos quanto nos rodapés.

Apêndice F Capturando Erros

Além de erros no conteúdo da tua documentação, existem dois tipos de erros que você pode cometer com o Texinfo: você pode cometer erros com os comandos @; e pode cometer erros com a estrutura dos nós e dos capítulos.

O Emacs tem duas ferramentas para detectar erros de comando @ e duas para detectar erros de estruturação.

Para encontrar problemas com os comandos @, você pode executar TEX ou um comando de formatação de região na região que tenha um problema; na verdade, você pode executar esses comandos em cada região conforme você a escreve.

Para encontrar problemas com a estrutura de nós e capítulos, você pode usar *C-c C-s* (`texinfo-show-structure`) e o comando relacionado `occur`, e você pode usar o comando *M-x Info-validate*.

F.1 `makeinfo` Preferido

O programa `makeinfo` faz um excelente trabalho de capturar erros e informá-los—muito melhor que `texinfo-format-region` ou `texinfo-format-buffer`. Além disso, as várias funções para criar e atualizar automaticamente ponteiros de nó e menus removem muitas oportunidades de erro humano.

Se puder, use os comandos de atualização para criar e inserir ponteiros e menus. Eles evitam muitos erros. A seguir use `makeinfo` (ou as manifestações dele do modo Texinfo, `makeinfo-region` e `makeinfo-buffer`) para formatar teu arquivo e verificar outros erros. Essa é a melhor maneira de trabalhar com o Texinfo. Mas se não puder usar `makeinfo`, ou se teu problema for muito intrigante, então você pode querer usar as ferramentas descritas neste anexo.

F.2 Detectando Erros com Formatação Info

Depois que tiver escrito parte de um arquivo do Texinfo, você pode usar o comando `texinfo-format-region` ou `makeinfo-region` para ver se a região formata corretamente.

Provavelmente, no entanto, você está lendo esta seção porque, por algum motivo, não pode usar o comando `makeinfo-region`; portanto, o restante desta seção presume que você está usando `texinfo-format-region`.

Se você tiver cometido um erro com um comando @, `texinfo-format-region` interromperá o processamento no erro ou depois dele e exibirá uma mensagem de erro. Para ver onde no buffer o erro ocorreu, comute para o buffer '`*Info Region*`'; o cursor estará em uma posição que é depois do local do erro. Além disso, o texto não será formatado depois do local onde o erro ocorreu (ou mais precisamente, onde ele foi detectado).

Por exemplo, se acidentalmente terminar um menu com o comando `@end menus` com um 's' no final, em vez de com `@end menu`, você verá uma mensagem de erro que diz:

```
@end menus não é manuseado pelo texinfo
```

O cursor irá parar no ponto no buffer onde o erro ocorre, ou não muito depois dele. O buffer se parecerá com isto:

```
---------- Buffer: *Info Region* ----------
* Menu:

* Usando texinfo-show-structure:: Como usar `texinfo-show-structure' para detectar erro
* Running Info-validate::         Como verificar nós não referenciados.
@end menus
*
---------- Buffer: *Info Region* ----------
```

O comando `texinfo-format-region` ocasionalmente fornece mensagens de erro ligeiramente estranhas. Por exemplo, a seguinte referência cruzada falha para formatar:

```
(@xref{Capturando Erros, para mais informações.)
```

Nesse caso, `texinfo-format-region` detecta a chave de fechamento ausente, mas exibe uma mensagem que diz '`Parênteses desbalanceados`' em vez de '`Chaves desbalanceadas`'. Isso ocorre porque o comando de formatação procura incompatibilidades entre chaves como se fossem parênteses.

Ocasionalmente, `texinfo-format-region` falha em detectar erros. Por exemplo, no seguinte, a chave de fechamento é trocada pelo parêntese de fechamento:

```
(@xref{Capturando Erros), para mais informações.}
```

A formatação produz:

```
(*Observação para mais informações: Capturando Erros)
```

A única maneira para você detectar esse erro é a de perceber que a referência deveria ter a seguinte aparência:

```
(*Observação Capturando Erros::, para mais informações)
```

Aliás, se você estiver lendo este nó no Info e digitar *`f RET`* (`Info-follow-reference`), você gerará uma mensagem de erro que diz:

```
Nenhum nó desse tipo: "Capturando Erros) A única maneira ...
```

Isso ocorre porque o Info percebe o exemplo do erro como a primeira referência cruzada nesse nó e se você digitar um `RET` imediatamente depois de digitar o comando *`f`* do Info, o Info tentará ir para o nó referenciado. Se você digitar *`f catch TAB RET`*, o Info completará o nome do nó do exemplo escrito corretamente e te levará para o nó 'Capturando Erros'. (Se você tentar isso, poderá retornar do nó 'Capturando Erros' digitando *`l`* (`Info-last`)).

F.3 Depuração com TEX

Você também pode detectar erros ao formatar um arquivo com TEX.

Normalmente, você vai querer fazer isso depois de ter executado `texinfo-format-buffer` (ou, melhor, `makeinfo-buffer`) no mesmo arquivo, porque `texinfo-format-buffer` ocasionalmente exibe mensagens de erro que fazem mais sentido que TEX. (Veja Seção F.2 [Depuração com Info], Página 252, para mais informações).

Por exemplo, TEX foi executado sobre um arquivo do Texinfo, parte do qual é mostrado aqui:

```
---------- Buffer: texinfo.texi ----------
nome do arquivo do Texinfo como uma extensão. Os @samp{??} são `curingas'
que fazem o shell substituir todos os arquivos de índices brutos.
(@xref{ordenação de índices, para mais informações acerca de ordenação de
índices).@refill
---------- Buffer: texinfo.texi ----------
```

(A referência cruzada carece de uma chave de fechamento). TEX produziu a seguinte saída, depois da qual parou:

```
---------- Buffer: *tex-shell* ----------
Argumento descontrolado?
{ordenação de índices, para mais informações acerca de ordenação de índices).
@refill @ETC.
! Parágrafo terminou antes que @xref estive completa.
<para ser lido novamente>
                   @par
l.27
```

```
?
---------- Buffer: *tex-shell* ----------
```

Nesse caso, TEX produziu uma mensagem de erro precisa e compreensível:

```
Parágrafo terminou antes que @xref estive completa.
```

'`@par`' é um comando interno do TEX sem relevância para o Texinfo. '`l.27`' significa que o TEX detectou o problema na linha 27 do arquivo do Texinfo. O '`?`' é o prompt que o TEX usa nessa circunstância.

Infelizmente, TEX nem sempre é tão útil, e ocasionalmente você precisa ser realmente um Sherlock Holmes para descobrir o que deu errado.

Em qualquer caso, se enfrentar um problema como esse, você pode fazer uma destas três coisas.

1. Você pode dizer ao TEX para continuar executando e ignorar apenas esse erro digitando `RET` no prompt '`?`'.
2. Você pode dizer ao TEX para continuar executando e ignorar todos os erros da melhor forma possível digitando *`r RET`* no prompt '`?`'.

   Geralmente, essa é a melhor coisa a fazer. No entanto, cuidado: um erro pode produzir uma cascata de mensagens de erro adicionais, pois as consequências dele são sentidas por todo o restante do arquivo. Para parar o TEX quando ele estiver produzindo uma avalanche de mensagens de erro, digite *`C-c`* (ou *`C-c C-c`*, se estiver executando um shell dentro do Emacs).
3. Você pode dizer ao TEX para interromper esta execução digitando *`x RET`* no prompt '`?`'.

Se você estiver executando o TEX dentro do Emacs, precisará comutar para o buffer do shell e a linha na qual o TEX oferece o prompt '`?`'.

Ocasionalmente, o TEX formatará um arquivo sem produzir mensagens de erro, mesmo que exista um problema. Isso geralmente ocorre se um comando não for finalizado, mas o TEX for capaz de continuar o processamento de qualquer maneira. Por exemplo, se você não conseguir finalizar uma lista itemizada com o comando `@end itemize`, o TEX escreverá um arquivo DVI que você consegue imprimir. A única mensagem de erro que o TEX te fornecerá é o comentário um tanto misterioso:

```
(@end ocorreu dentro de um grupo no nível 1)
```

No entanto, se você imprimir o arquivo DVI, verá que o texto do arquivo que segue a lista itemizada está inteiramente recuado como se fosse parte do último item na lista itemizada. A mensagem de erro é a maneira como o TEX diz que esperava encontrar um comando `@end` em algum lugar no arquivo; mas que não poderia determinar onde ele era necessário.

Outra fonte de erros notoriamente difíceis de encontrar é um comando `@end group` ausente. Se você alguma vez ficar perplexo com erros incompreensíveis, procure por um comando `@end group` ausente primeiro.

Se o arquivo do Texinfo carecer de linhas de cabeçalho, TEX pode parar no começo da execução e exibir uma saída que se parece com a seguinte. O '`*`' indica que TEX está esperando por entrada.

```
Este é o TeX, Versão 3.14159 (Web2c 7.0)
(test.texinfo [1])
*
```

Nesse caso, simplesmente digite *`\end RET`* depois do asterisco. A seguir escreva as linhas de cabeçalho no arquivo do Texinfo e execute o comando TEX novamente. (Observe o uso da barra invertida, '`\`'. TEX usa '`\`' em vez de '`@`'; e nessa circunstância, você está trabalhando diretamente com TEX, não com Texinfo).

F.4 Usando `texinfo-show-structure`

Nem sempre é fácil manter o controle de nós, capítulos, seções e subseções de um arquivo do Texinfo. Isso é especialmente verdadeiro se você estiver revisando ou adicionando a um arquivo do Texinfo que outra pessoa tenha escrito.

No GNU Emacs, no modo Texinfo, o comando `texinfo-show-structure` lista todas as linhas que começam com os comandos @ que especificam a estrutura: `@chapter`, `@section`, `@appendix` e assim por diante. Com um argumento (*C-u* como argumento de prefixo, se interativo), o comando também mostra as linhas `@node`. O comando `texinfo-show-structure` é vinculado a *C-c C-s* no modo Texinfo, por padrão.

As linhas são exibidas em um buffer chamado '`*Occur*`', recuado por nível hierárquico. Por exemplo, aqui está uma parte do que foi produzido ao executar `texinfo-show-structure` sobre este manual:

```
Linhas correspondentes a "^@\\(chapter \\|sect\\|subs\\|subh\\|
unnum\\|major\\|chapheading \\|heading \\|appendix\\)"
no buffer texinfo.texi.
...
4177:@chapter Nós
4198:    @heading Dois Caminhos
4231:    @section Ilustração de Nó e Menu
4337:    @section O Comando @code{@@node}
4393:        @subheading Escolhendo Nomes de Nós e Ponteiros
4417:        @subsection Como Escrever uma Linha @code{@@node}
4469:        @subsection Dicas da Linha @code{@@node}
...
```

Isso diz que as linhas 4337, 4393 e 4417 de `texinfo.texi` começam com os comandos `@section`, `@subheading` e `@subsection` respectivamente. Se você mover teu cursor para a janela '`*Occur*`', você pode posicionar o cursor sobre uma das linhas e usar o comando *C-c C-c* (`occur-mode-goto-occurrence`), para pular para o ponto correspondente no arquivo do Texinfo. Veja Seção "Using Occur" em *O Manual do GNU Emacs*, para mais informações acerca de `occur-mode-goto-occurrence`.

A primeira linha na janela '`*Occur*`' descreve a *expressão regular* especificada por *texinfo-heading-pattern*. Essa expressão regular é o padrão que `texinfo-show-structure` procura. Veja Seção "Using Regular Expressions" em *O Manual do GNU Emacs*, para mais informações.

Quando você invoca o comando `texinfo-show-structure`, o Emacs exibirá a estrutura do buffer inteiro. Se você quiser ver a estrutura de apenas uma parte do buffer, de um capítulo, por exemplo, use o comando *C-x n n* (`narrow-to-region`) para marcar a região. (Veja Seção "Narrowing" em *O Manual do GNU Emacs*). É assim que o exemplo usado acima foi gerado. (Para ver o buffer inteiro novamente, use *C-x n w* (`widen`)).

Se você chamar `texinfo-show-structure` com um argumento de prefixo digitando *C-u C-c C-s*, ele listará as linhas começando com `@node`, bem como as linhas começando com os comandos de sinal @ para `@chapter`, `@section` e similares.

Você pode se lembrar da estrutura de um arquivo do Texinfo observando a lista na janela '`*Occur*`'; e se tiver nomeado um nó incorretamente ou deixou uma seção de fora, você consegue corrigir o erro.

F.5 Usando `occur`

Ocasionalmente, o comando `texinfo-show-structure` produz muita informação. Talvez você queira se lembrar da estrutura geral de um arquivo do Texinfo e esteja sobrecarregado(a) pela

lista detalhada produzida por `texinfo-show-structure`. Nesse caso, você pode usar o comando `occur` diretamente. Para fazer isso, digite:

```
M-x occur
```

e então, quando solicitado(a), digite *regexp*, uma expressão regular para o padrão que você quer corresponder. (Veja Seção "Regular Expressions" em *O Manual do GNU Emacs*). O comando `occur` funciona a partir da localização atual do cursor no buffer até o fim do buffer. Se você quiser executar `occur` sobre o buffer inteiro, coloque o cursor no começo do buffer.

Por exemplo, para ver todas as linhas que contenham a palavra '`@chapter`', basta digitar '`@chapter`'. Isso produzirá uma lista dos capítulos. Também listará todas as frases com '`@chapter`' no meio da linha.

Se você quiser ver somente aquelas linhas que começam com a palavra '`@chapter`', digite '`^@chapter`' quando solicitado(a) por `occur`. Se você quiser ver todas as linhas que terminam com uma palavra ou frase, termine a última palavra com '`$`'; por exemplo, '`capturando erros$`'. Isso pode ser útil quando você quiser ver todos os nós que sejam parte do mesmo capítulo ou seção e, portanto, tem o mesmo ponteiro 'Acima'.

Veja Seção "Using Occur" em *O Manual do GNU Emacs*, para mais informações.

F.6 Encontrando Nós Mal Referenciados

Você pode usar o comando `Info-validate` para verificar se algum dos ponteiros de nó 'Próximo', 'Anterior', 'Acima' ou outros falham em apontar para um nó. Esse comando verifica se cada ponteiro de nó aponta para um nó existente. O comando `Info-validate` funciona somente sobre arquivos do Info, não sobre arquivos do Texinfo.

O programa `makeinfo` valida ponteiros automaticamente, de forma que você não precisa usar o comando `Info-validate` se estiver usando `makeinfo`. Você só pode precisar usar `Info-validate` se não conseguir executar `makeinfo` e, em vez disso, precisar criar um arquivo do Info usando `texinfo-format-region` ou `texinfo-format-buffer`, ou se escrever um arquivo do Info desde o zero.

F.6.1 Usando `Info-validate`

Para usar `Info-validate`, visite o arquivo do Info que deseja verificar e digite:

```
M-x Info-validate
```

Observe que o comando `Info-validate` exige um 'I' maiúsculo. Você também pode precisar criar uma tabela de etiquetas antes de executar `Info-validate`. Veja Seção F.6.3 [Etiquetando], Página 257.

Se teu arquivo for válido, você receberá uma mensagem que diz "Arquivo parece válido". No entanto, se você tiver um ponteiro que não aponta para um nó, mensagens de erro serão exibidas em um buffer chamado '`*problemas no arquivo info*`'.

Por exemplo, `Info-validate` foi executado sobre um arquivo de teste que continha somente o primeiro nó deste manual. Uma das mensagens dizia:

```
No nó "Visão Geral", inválido Próximo: Modo Texinfo
```

Isso significava que o nó chamado '`Visão Geral`' tinha um ponteiro 'Próximo' que não apontava para nada (o que era verdade nesse caso, já que o arquivo de teste tinha somente um nó).

Agora, suponha que nós adicionamos um nó chamado '`Modo Texinfo`' ao nosso caso de teste, mas não especificamos um 'Anterior' para esse nó. Então, obteremos a seguinte mensagem de erro:

```
No nó "Modo Texinfo", deveria ter Anterior: Visão Geral
```

Isso ocorre porque cada ponteiro 'Próximo' deveria ser correspondido por um 'Anterior' (no nó para onde o 'Próximo' aponta), que aponta para trás.

`Info-validate` também verifica se todas as entradas de menu e referências cruzadas apontam para nós reais.

`Info-validate` exige uma tabela de etiquetas e não funciona com arquivos que tenham sido divididos. (O comando `texinfo-format-buffer` divide automaticamente arquivos grandes). Para a finalidade de usar `Info-validate` sobre um arquivo grande, você precisa executar `texinfo-format-buffer` com um argumento, de forma que ele não divida o arquivo do Info; e você precisa criar uma tabela de etiquetas para o arquivo não dividido.

F.6.2 Criando um Arquivo Desdividido

Você pode executar `Info-validate` somente sobre um arquivo unitário do Info que tenha uma tabela de etiquetas. O comando não funcionará sobre os sub arquivos indiretos que sejam gerados quando um arquivo mestre for dividido. Se você tiver um arquivo grande (maior que 300.000 bytes), você precisa executar o comando `texinfo-format-buffer` ou `makeinfo-buffer` de tal forma que ele não crie subarquivos indiretos. Você também precisará criar uma tabela de etiquetas para o arquivo do Info. Depois de fazer isso, você pode executar `Info-validate` e procurar por nós referenciados incorretamente.

O primeiro passo é o de criar um arquivo do Info desdividido. Para evitar que o `texinfo-format-buffer` divida um arquivo do Texinfo em arquivos do Info menores, forneça um prefixo para o comando *`M-x texinfo-format-buffer`*:

```
C-u M-x texinfo-format-buffer
```

ou então

```
C-u C-c C-e C-b
```

Ao fazer isso, o Texinfo não dividirá o arquivo e não criará uma tabela de etiquetas para ele.

F.6.3 Etiquetando um Arquivo

Depois de criar um arquivo do Info desdividido, você precisa criar uma tabela de etiquetas para ele. Visite o arquivo do Info que você deseja etiquetar e digite:

```
M-x Info-tagify
```

(Observe a letra maiúscula '`I`' no `Info-tagify`). Isso cria um arquivo do Info com uma tabela de etiquetas que você consegue validar.

O terceiro passo é o de validar o arquivo do Info:

```
M-x Info-validate
```

(Observe a letra maiúscula '`I`' no `Info-validate`). Resumidamente, as etapas são:

```
C-u M-x texinfo-format-buffer
M-x Info-tagify
M-x Info-validate
```

Depois que tiver validado a estrutura do nó, você pode executar novamente `texinfo-format-buffer` da maneira normal, de forma que ele construirá uma tabela de etiquetas e dividirá o arquivo automaticamente; ou você pode criar a tabela de etiquetas e dividir o arquivo manualmente.

F.6.4 Dividindo um Arquivo Manualmente

Você deveria dividir um arquivo grande ou então deixar que o comando `texinfo-format-buffer` ou `makeinfo-buffer` faça isso para você automaticamente. (Geralmente você deixará um dos comandos de formatação fazer esse trabalho para você. Veja Seção 21.1 [Criando um Arquivo do Info], Página 185).

Os arquivos divididos são chamados de subarquivos indiretos.

Arquivos do Info são divididos para economizar memória. Com arquivos menores, o Emacs não precisa criar um buffer tão grande para manter as informações.

Se um arquivo do Info tiver mais de 30 nós, você também deveria criar uma tabela de etiquetas para ele. Veja Seção F.6.1 [Usando `Info-validate`], Página 256, para informações acerca de criar uma tabela de etiquetas. (Novamente, as tabelas de etiquetas geralmente são criadas automaticamente pelo comando de formatação; você só precisa criar uma tabela de etiquetas se estiver fazendo o trabalho manualmente. Provavelmente, você fará isso para um arquivo grande e não dividido sobre o qual tenha executado `Info-validate`).

Visite o arquivo do Info que você deseja etiquetar e dividir e digite os dois comandos:

```
M-x Info-tagify
M-x Info-split
```

(Observe que o '`I`' em '`Info`' é maiúsculo).

Quando você usa o comando `Info-split`, o buffer é modificado em um (pequeno) arquivo do Info que lista os subarquivos indiretos. Esse arquivo deveria ser salvo no lugar do arquivo original visitado. Os subarquivos indiretos são escritos no mesmo diretório em que o arquivo original está, com nomes gerados anexando '`-`' e um número ao nome do arquivo original.

O arquivo principal ainda funciona como um arquivo do Info, mas contém somente a tabela de etiquetas e um diretório de subarquivos.

Apêndice G Especificação do Formato Info

Aqui nós descrevemos os detalhes técnicos do formato do Info.

Nesta descrição formal, os caracteres `<>*()|=#` são usados para o idioma da descrição em si. Outros caracteres são literais. As construções formais usadas são típicas: `<...>` indica um nome de meta variável, '`=`' significa definição, '`*`' repetição, '`?`' opcional, '`()`' agrupamento, '`|`' alternância, '`#`' comentário. Exceção: '`*`' no início de uma linha é literal.

Em geral, programas que leem arquivos do Info deveriam tentar não diferenciar maiúsculas de minúsculas de palavras-chave que ocorrerem no arquivo (por exemplo, '`Tabela de Etiquetas`' e '`Tabela de etiquetas`' deveriam ser equivalentes) para a finalidade de suportar programas geradores de Info que usem capitalização diferente.

As seções em um arquivo do Info (como nós ou tabelas de etiquetas) são separadas com uma sequência:

```
(^L)?^_(^L)?^J
```

Isto é, um caractere '`CTRL-_`' seguido por uma nova linha, com caracteres de avanço de formulário opcionais. Nós nos referimos a tais sequências como `<separador>`.

Nós especificamos parênteses literais (aqueles que são parte do formato do Info) com `<lparen>` e `<rparen>`, significando os caracteres unitários '`(`' e '`)`' respectivamente. Nós especificamos o caractere '`CTRL-?`' (número de caractere 127) `<del>`. Finalmente, a sequência de dois caracteres '`^x`' significa o caractere unitário '`CTRL-x`', para qualquer *x*.

Esta definição de formato foi escrita cerca de 25 anos depois que o formato do Info foi concebido. Portanto, no caso de conflitos entre esta definição e a prática real, a prática vence. Ela também pressupõe algum conhecimento geral do Texinfo; ela foi criada para ser um guia para implementadores(as), em vez de um padrão técnico rígido. Frequentemente, nós recorremos a outras partes deste manual para exemplos e definições, em vez de soletrar redundantemente cada detalhe.

G.1 Esquema Geral do Formato do Info

Esta seção descreve o esquema geral dos manuais do Info.

Formato do Info: Um Manual Inteiro

Para começar, um manual do Info é ou *não dividido* (contido inteiramente em um arquivo) ou *dividido* (em vários arquivos).

A sintaxe para um manual não dividido é:

```
    <arquivo não dividido do info> =
<preâmbulo>
<nó>*
<tabela de etiquetas>?
<variáveis locais>?
```

Quando dividido, existe um *arquivo principal*, que contém somente ponteiros para os nós fornecidos em outros *subarquivos*. O arquivo principal se parece com isto:

```
    <arquivo principal dividido do info> =
<preâmbulo>
<tabela indireta>
<tabela de etiquetas>
<variáveis locais>?
```

Os subarquivos em um manual dividido tem a seguinte sintaxe:

```
    <subarquivo dividido do info> =
```

```
<preâmbulo>
<nó>*
```

Observe que a tabela de etiquetas não é opcional para arquivos divididos, pois ela é usada com a tabela indireta para deduzir em qual subarquivo um nó específico está.

Formato do Info: Preâmbulo

O `<preâmbulo>` é um texto no começo de todos os arquivos de saída. Ele não é destinado a ser visível por padrão em um visualizador do Info, mas pode ser exibido mediante solicitação do(a) usuário(a).

```
    <preâmbulo> =
<identificação>             # "Este é NOMEARQUIVO, produzido por ..."
<texto da copiagem>         # Expansão do texto do @copying.
<entradas de diretórios>    # Derivadas de @dircategory e @direntry.
```

Esses pedaços são:

`<linha de identificação>`
: Uma sequência arbitrária iniciando o arquivo de saída, seguida por uma linha em branco.

`<texto da copiagem>`
: A expansão de um ambiente `@copying`, se o manual tiver um (veja Seção 3.3.1 [`@copying`], Página 17).

`<entradas de diretórios>`
: O resultado de quaisquer comandos `@dircategory` e `@direntry` presentes no manual (veja Seção 21.2.4 [Instalando Entradas de Diretório], Página 190).

Formato do Info: Tabela Indireta

```
    <tabela indireta> =
<separador>
Indireta:
(<nomearquivo>: <posiçãobyte>)*
```

A tabela indireta é escrita no arquivo principal somente no caso de saída dividida. Ela especifica, como um inteiro decimal, a posição inicial do byte (baseada em zero) que o primeiro nó de cada subarquivo teria se os subarquivos fossem concatenados em ordem, não incluindo o arquivo de nível superior. O primeiro nó do conteúdo real é apontado pela primeira entrada.

Como exemplo, suponha que a saída dividida seja gerada para o manual do GDB. O arquivo de nível superior `gdb.info` conterá algo como isto:

```
<separador>
Indireta:
gdb.info-1: 1878
gdb.info-2: 295733
...
```

Isso informa aos visualizadores do Info que o primeiro nó do manual ocorre no byte 1878 do arquivo `gdb.info-1` (que estaria depois do preâmbulo desse arquivo). O primeiro nó no subarquivo `gdb.info-2` começaria no byte 295733 se `gdb.info-2` fosse posposto a `gdb.info-1`, incluindo quaisquer seções de preâmbulo em ambos os arquivos.

Infelizmente, programas de criação do Info, como o `makeinfo`, nem sempre implementaram essas regras perfeitamente, devido a vários defeitos e descuidos. Portanto, visualizadores robustos do Info deveriam voltar a procurar "próximo" à posição fornecida para um nó, em vez de desistir imediatamente se a posição não estiver exatamente no início de um nó.

Formato do Info: Tabela de Etiquetas

```
    <tabela de etiquetas> =
<separador>
Tabela de Etiquetas:
(<lparen>Indireta<rparen>)?
(Nó|Ref): <idnó>^?<posiçãobyte>
<separador>
Fim Tabela de Etiquetas
```

A linha '`(Indireta)`' aparece somente no caso de saída dividida.

A tabela de etiquetas especifica a posição inicial do byte de cada nó e âncora no arquivo. No caso de saída dividida, ela é escrita somente no arquivo de saída principal.

Cada linha define um identificador como ou uma âncora ou um nó, conforme especificado. Por exemplo, '`Node: Top^?1647`' diz que o nó chamado '`Top`' começa no byte 1647, enquanto '`Ref: Overview-Footnote-1^?30045`' diz que a âncora chamada '`Overview-Footnote-1`' começa no byte 30045. É um erro definir o mesmo identificador em ambas as formas.

No caso de saída não dividida, as posições de byte simplesmente se referem ao local no arquivo de saída. No caso de saída dividida, as posições de byte se referem a um arquivo imaginário criado pela concatenação de todos os arquivos divididos (mas não o arquivo de nível superior). Veja-se a seção anterior.

Aqui está um exemplo:

```
^_
Tabela de Etiquetas:
Nó: Top^?89
Nó: Ch1^?292
^_
Fim Tabela de Etiquetas
```

Isso especifica um manual com dois Nós, 'Top' e 'Ch1', nas posições de byte 89 e 292, respectivamente. Como a linha '`(Indireta)`' não está presente, o manual não é dividido.

Seções de preâmbulo ou outras seções de arquivos não nós não tem uma entrada na tabela de etiquetas.

Formato do Info: Variáveis Locais

A seção de variáveis locais é opcional e é usada atualmente para fornecer as informações de codificação. Ela pode ser aumentada no futuro.

```
    <variáveis locais> =
<separador>
Variáveis Locais:
codificação: <codificação>
End:
```

Veja Seção 15.2 [`@documentencoding`], Página 126.

Formato do Info: Nós Regulares

Nós regulares se parecem com isto:

```
    <nó> =
<separador>
Arquivo: <fn>, Nó: <id1>, (Próximo: <id2>, )? (Anterior: <id3>, )? Acima: <id4>

<texto geral, até o próximo ^_ ou fim-do-arquivo>
```

Pelo menos um espaço ou tabulação precisa estar presente depois de cada dois pontos e vírgula, mas qualquer número de espaços é ignorado. Os identificadores de nó `<id>` tem o seguinte formato:

```
      <id> = (<lparen><arquivoinfo><rparen>)?(<del>?<nomenó><del>?)?
  |   <id> = (<lparen><arquivoinfo><rparen>)?(<nomenó>)?
```

Esse `<nó>` define `<id1>` no arquivo `<fn>`, que normalmente é ou '`nomemanual`' ou '`nomemanual.info`'. Nenhum componente `<arquivoinfo>` entre parênteses pode aparecer dentro de `<id1>`.

Cada um dos identificadores depois de `Próximo`, `Anterior` e `Acima` se refere a nós ou âncoras dentro de um arquivo. Esses ponteiros normalmente referenciam dentro do mesmo arquivo, mas '`(dir)`' frequentemente é usado para apontar para o arquivo Dir de nível superior. Se um componente `<arquivoinfo>` for usado, então o nome do nó pode ser omitido, caso no qual o identificador do nó se refere ao nó '`Top`' dentro do arquivo referenciado.

Os ponteiros `Próximo` e `Anterior` são opcionais. O ponteiro `Acima` tecnicamente também é opcional, embora muito provavelmente isso indique um erro na estruturação do nó. Convencionalmente, os nós são organizados para formar uma árvore, mas isso não é uma exigência do formato.

Nomes de nós contendo pontos, vírgulas, dois pontos ou parênteses (incluindo comandos @ que produzem qualquer um desses) podem confundir leitores do Info. Se for necessário se referir a um nó cujo nome contenha qualquer um desses, o `<nomenó>` deveria ser cercado por um par de caracteres `<del>`. Existe suporte no `makeinfo` para adicionar esses caracteres (veja [INFO_SPECIAL_CHARS_QUOTE], Página 178); no entanto, nós não recomendamos que você faça uso desse suporte até que programas de leitura do Info que reconheçam essa sintaxe sejam comuns. Veja Seção 4.4 [Exigências de Linha de Nó], Página 31.

O uso de caracteres não-ASCII nos nomes de nós é permitido, mas pode causar problemas em referências cruzadas entre nós em arquivos do Info com diferentes codificações de caracteres, e também quando nomes de nós provenientes de muitos arquivos forem listados (por exemplo, com a opção `--apropos` para o navegador autônomo do Info), de forma que nós recomendamos evitá-los sempre que possível. Por exemplo, prefira o uso do caractere de apóstrofo do ASCII (') às aspas direcionais do Unicode.

O `<texto geral>` do nó pode incluir as construções especiais descritas a seguir.

G.2 Construtores de Texto do Formato do Info

Essas construções especiais do Info podem aparecer dentro do texto de um nó.

G.2.1 Formato do Info: Menu

Convencionalmente, os menus aparecem no final dos nós, mas o formato do Info não impõe restrições quanto à localização deles.

```
      <menu> =
* Menu:
(<entrada de menu> | <comentário de menu>)*
```

As partes de uma `<entrada de menu>` também estão descritas em Seção 4.9.4 [Partes de Menu], Página 37. Elas tem a mesma sintaxe que referências cruzadas (veja Seção G.2.4 [Referências Cruzadas do Formato do Info], Página 263). Índices estendem o formato de menu para especificar a linha de destino; veja Seção G.2.3 [Imprime índices do Formato do Info], Página 263.

Um `<comentário de menu>` é qualquer linha não iniciando com '`*`' que aparece ou no início do menu ou é separada de uma entrada de menu por uma ou mais linhas em branco. Esses comentários são destinados a serem exibidos como parte do menu, como está (veja Seção 4.9.1 [Escrevendo um Menu], Página 36).

G.2.2 Formato do Info: Imagem

O comando `@image` resulta na seguinte diretiva especial dentro do arquivo do Info (veja Seção 10.2 [Imagens], Página 84):

```
     <imagem> =
^@^H[imagem src="<arquivo de imagem>"
            (texto="<conteúdo do arquivo de texto>")?
            (alt="<texto alternativo>")?
^@^H]
```

As quebras de linha e recuos nesta descrição são editoriais; o espaço em branco entre as diferentes partes da diretiva nos arquivos do Info é arbitrário.

Nas strings `<arquivo de imagem>`, `<conteúdo do arquivo de texto>` e `<texto alternativo>`, '`"`' é aspado como '`\"`' e '`\`' é aspado como '`\\`'. As especificações de texto e alt são opcionais.

O valor de `alt` serve ao mesmo propósito que no HTML: Uma descrição em prosa da imagem. Em exibições somente de texto ou em sistemas de fala, por exemplo, o valor de `alt` pode ser usado em vez de exibir o (tipicamente gráfico) `<arquivo de imagem>`.

O `<conteúdo do arquivo de texto>`, se presente, deveria ser considerado uma representação ASCII da imagem, para possível uso em uma exibição somente de texto.

O formato não prescreve a escolha entre exibir o `<arquivo de imagem>`, o `<texto alternativo>` ou o `<conteúdo do arquivo de texto>`.

G.2.3 Formato do Info: Imprime índices

Índices no formato do Info geralmente são escritos como um menu (veja Capítulo 11 [Índices], Página 89), mas com uma diretiva adicional no início marcando esse como um nó de índice:

```
     <imprimeíndice> =
^@^H[índice^@^H]
* Menu:

<entrada de índice>*
```

Os itens `<entrada de índice>` são semelhantes às entradas normais de menu, mas a descrição de formato livre é substituída pelo número da linha onde as entradas ocorrem no texto:

```
     <entrada de índice> =
* <texto de entrada>: <nó de entrada>. <lparen>linha <númerolinha><rparen>
```

O `<texto de entrada>` é o termo de índice. O `<númerolinha>` é um inteiro não sinalado, fornecido relativo ao início de `<nó de entrada>`. Podem existir espaços em branco arbitrários depois dos dois pontos e o do ponto, como de costume em menus, e podem existir quebras nas linhas. Aqui está um exemplo:

```
^@^H[índice^@^H]
* Menu:

* trovão:          Fenômenos Meteorológicos.            (line 5)
```

Isso significa que uma entrada de índice para 'trovão' aparece na linha 5 do nó 'Fenômenos Meteorológicos'.

G.2.4 Formato do Info: Referência Cruzada

Uma referência cruzada geral no formato do Info tem um dos seguintes dois formatos:

```
     <referência-cruzada> =
  * (N|n)ote <id>::
```

```
| * (N|n)ote <rótulo>:<id>(.|,)

     <id> = (<lparen><arquivoinfo><rparen>)?(<del>?<nomenó><del>?)?
|    <id> = (<lparen><arquivoinfo><rparen>)?(<nomenó>)?
    <rótulo> = <del>?<texto do rótulo><del>?
```

Nenhum espaço deveria ocorrer entre o caractere '`*`' e o seguinte '`N`' ou '`n`'. '`*Note`' deveria ser usado no início de uma frase, caso contrário, '`*note`' deveria ser usado. (Alguns leitores do Info, como o do Emacs, podem exibir '`*Note`' e '`*note`' como '`See`' e '`see`', respectivamente). Em ambos os casos, `<texto do rótulo>` é um texto descritivo.

Em ambas as formas, o `<id>` referencia um nó ou âncora, da mesma maneira que uma referência na linha de informações do nó faz (veja [Nós Regulares do Formato do Info], Página 261). O '`<arquivoinfo>`' opcional entre parênteses é o nome do arquivo do manual sendo referenciado, e o `<nomenó>` é o nó ou âncora dentro desse manual.

A segunda forma tem um rótulo descritivo. Uma referência cruzada nessa forma deveria geralmente ser terminada com uma vírgula ou um ponto, para tornar viável encontrar o fim do `<id>`.

Se `<rótulo>` contiver um caractere de dois pontos (`:`), ele deveria ser cercado por um par de caracteres `<del>`. Da mesma forma, se `<nomenó>` contiver caracteres problemáticos (como vírgulas ou pontos), ele deveria ser cercado por um par de caracteres `<del>`; então uma vírgula ou um ponto de terminação não é necessária.

Assim como com os nomes de nós, esse mecanismo de aspeamento tem, até o momento em que este texto foi escrito, suporte limitado em programas de leitura do Info; portanto, nós não recomendamos usá-lo até que isso mude.

O formato não prescreve como encontrar outros manuais para resolver tais referências.

Aqui estão alguns exemplos:

```
*note Licença GNU de Documentação Livre::
*note Tabela de etiquetas: Tabela de Etiquetas do Formato do Info, para detalhes.
*Note Visão Geral: (make)Top.
*Note ^?:^?: (bash)Bourne Shell Builtins.
*Note alloca.h: (gnulib)^?alloca.h^?.
```

O primeiro mostra uma referência a um nó no manual atual usando o formato curto.

O segundo também referencia um nó no manual atual, chamado 'Tabela de Etiquetas do Formato do Info'; a 'Tabela de etiquetas' antes de '`:`' é somente um rótulo nessa referência específica, e '`para detalhes.`' é um texto pertencente à frase, não parte da referência.

O terceiro exemplo referencia o nó 'Top' em outro manual, chamado '`make`', com 'Visão Geral' sendo o rótulo para essa referência cruzada.

O quarto exemplo mostra um caractere de dois pontos sendo aspado em um rótulo, e o quinto exemplo mostra um ponto sendo aspado em um nome de nó. Veja Capítulo 6 [Referências Cruzadas], Página 45.

Apêndice H Licença GNU de Documentação Livre

Versão 1.3, 03 de novembro de 2008

0. PREÂMBULO

O propósito desta licença é tornar um manual, livro de texto, ou outro documento funcional e útil *livre* no sentido da liberdade: para assegurar a qualquer pessoa a liberdade efetiva para copiar e redistribuí-lo, com ou sem modificações, ambos comercialmente ou não comercialmente. Secundariamente, esta Licença preserva para o autor e editor uma maneira de obter crédito pelos seus trabalhos, ao mesmo tempo não sendo considerado responsável por modificações feitas por outros.

Esta Licença é uma espécie de "copyleft" ("esquerdos autorais"), o que significa que trabalhos derivados do documento devem necessariamente eles mesmos serem livres no mesmo sentido. Ela complementa a Licença Pública Geral GNU, a qual é uma licença de esquerdos autorais projetada para software livre.

Nós projetamos esta Licença para utilizá-la para manuais para software livre, porque software livre precisa de documentação livre: um programa livre deveria vir com manuais provendo as mesmas liberdades que o software provê. Porém esta Licença não é limitada a manuais de software; ela pode ser utilizada para qualquer trabalho textual, independentemente de questões de assunto ou se o trabalho textual for publicado como um livro impresso. Nós recomendamos esta Licença principalmente para trabalhos cujo propósito seja instrução ou referência.

1. APLICABILIDADE E DEFINIÇÕES

Esta Licença se aplica a qualquer manual ou outro trabalho, em qualquer meio, que contenha um aviso colocado pelo detentor dos direitos autorais dizendo que ele pode ser distribuído sob os termos desta Licença. Tal aviso concede uma licença mundial, livre de patente, ilimitada na duração, para utilizar aquele trabalho sob as condições nela declaradas. O "Documento", abaixo, se refere a quaisquer desses manuais ou trabalhos. Qualquer membro do publico é um titular da licença, e é mencionado como "você". Você aceita a licença se você copiar, modificar ou distribuir o trabalho em uma forma que exija permissão sob lei de direitos autorais.

Uma "Versão Modificada" do Documento significa qualquer trabalho contendo o Documento ou uma porção dele, seja literalmente copiado, ou com modificações e/ou traduzido em outra língua.

Uma "Seção Secundária" é um apêndice nomeado ou uma seção pré-textual do Documento que lida exclusivamente com o relacionamento dos editores ou autores do Documento para com o assunto global do Documento (ou com questões relacionadas) e não contém nada que possa se conformar diretamente com aquele assunto global. (Assim, se o Documento for em parte um livro texto de matemática, uma Seção Secundária não pode explanar nada acerca de cálculos matemáticos). O relacionamento poderia ser uma questão de conexão histórica com o assunto ou com questões relacionadas, ou de posicionamento legal, comercial, filosófico, ético ou político respeitante a eles.

As "Seções Invariantes" são certas Seções Secundárias cujos títulos são projetados, como sendo aqueles de Seções Invariantes, no aviso que diz que o Documento é publicado sob esta Licença. Se uma seção não se encaixa na definição de Secundária acima, então a seção não está autorizada a ser designada como Invariante. O Documento pode conter zero Seções

Invariantes. Se o Documento não identifica quaisquer Seções Invariantes, então não existe nenhuma.

Os "Textos de Capa" são certas passagens curtas de texto que são listadas, como Textos de Primeira Capa ou Textos de Quarta-Capa, no aviso que diz que o Documento é publicado sob esta Licença. Um Texto de Primeira Capa pode ter no máximo cinco (05) palavras, e um Texto de Quarta Capa pode ter no máximo vinte e cinco (25) palavras.

Uma cópia "Transparente" do Documento significa uma cópia legível por máquina, representada em um formato cuja especificação está disponível para o público em geral, que é adequada para revisar o documento diretamente com editores de texto genéricos ou (para imagens compostas de pixeis) programas de pintura genéricos ou (para desenhos) algum editor de desenho disponível amplamente, e que seja adequado para entrada a formatadores de texto ou para tradução automática a uma variedade de formatos próprios para entrada a formatadores de texto. Uma cópia feita em um formato de arquivo contrário ao Transparente, cuja linguagem de marcação, ou ausência de linguagem de marcação, tenha sido organizada para frustrar ou desencorajar modificações subsequentes por leitores, não é Transparente. Um formato de imagem não é Transparente se utilizado para qualquer quantidade substancial de texto. Uma cópia que não é "Transparente" é chamada "Opaca".

Exemplos de formatos adequados para cópias Transparentes incluem ASCII puro sem marcações; formato de entrada Texinfo; formato de entrada LaTeX; SGML ou XML utilizando um DTD disponível publicamente; HTML simples conformante com o padrão; PostScript ou PDF projetado para modificação humana. Exemplos de formatos transparantes de imagens incluem PNG, XCF e JPG. Formatos opacos incluem formatos proprietários que podem ser lidos e editados somente por processadores proprietários de palavra; SGML ou XML para os quais o DTD e/ou as ferramentas de processamentos não estejam disponíveis genericamente; e o HTML gerado por máquina; PostScript ou PDF produzidos por alguns processadores de palavra apenas para propósitos de saída.

A "Página de Título" significa, para um livro impresso, a própria página de título, mais tantas páginas seguintes quantas sejam necessárias para manter, legivelmente, o material que esta Licença exige para aparecer na página de título. Para trabalhos em formatos que não tenham qualquer página de título como tal, "Página de Título" significa o texto próximo da mais proeminente aparição do título do trabalho, precedendo o início do corpo do texto.

O "editor" significa qualquer pessoa ou entidade que distribui cópias do Documento ao público.

Uma seção "Intitulada XYZ" significa uma subunidade nomeada do Documento cujo título ou é precisamente XYZ ou contém XYZ entre parênteses seguinte ao texto que traduz XYZ em outra linguagem. (Aqui XYZ significa um nome específico de seção mencionado abaixo, tais como "Agradecimentos"; "Dedicatórias"; "Patrocínios"; ou "Histórico"). "Preservar o Título" de tal seção quando você modificar o Documento significa que ele permanece uma seção "Intitulada XYZ" de acordo com essa definição.

O Documento pode incluir Declarações de Garantia próximas ao aviso que declara que esta Licença se aplica ao Documento. Essas Declarações de Garantia são consideradas como inclusas por referência nesta Licença, porém somente com relação à negação de garantias: qualquer outra implicação que essas Declarações de Garantia possam ter é inválida e não tem efeito sobre o significado desta Licença.

2. CÓPIA LITERAL

Você pode copiar e distribuir o Documento em qualquer meio, ambos comercialmente e não comercialmente, contanto que esta Licença, os avisos de direitos autorais, e o aviso de licença dizendo que esta Licença se aplica ao Documento estejam reproduzidas em todas as cópias, e que você não adiciona quaisquer outras condições, quaisquer que sejam, àquelas

desta Licença. Você não pode utilizar medidas técnicas para obstruir ou controlar a leitura ou posteriores cópias das cópias que você fizer ou distribuir. Entretanto, você pode aceitar remuneração em troca das cópias. Se você distribui um número de cópias grande o suficiente, você deve necessariamente também seguir as condições na seção três (3).

Você também pode ceder cópias, sob as mesmas condições declaradas acima, e você pode publicamente exibir cópias.

3. CÓPIAS EM QUANTIDADE

   Se você publicar cópias impressas (ou cópias em mídia que geralmente tem capas impressas) do Documento, em número maior que cem (100), e o aviso de licença do Documento exigir Textos de Capa, você deve necessariamente encartar as cópias em capas que transportem, claramente e legivelmente, todos estes Textos de Capa: Textos de Primeira Capa na primeira capa, e Textos de Quarta Capa na capa traseira. Ambas as capas devem necessariamente também claramente e legivelmente identificar você como o editor dessas cópias. A capa frontal deve necessariamente apresentar o título completo com todas as palavras do título igualmente proeminentes e visíveis. Você pode adicionar outros materiais nas capas adicionalmente. As cópias com modificações limitadas às capas, tanto quanto preservem o título do Documento e satisfaçam essas condições, podem ser tratadas como cópias literais em relação a outros aspectos.

   Se os textos exigidos para ambas as capas forem muito volumosos para caber legivelmente, você deveria colocar os primeiros listados (tantos quantos caibam razoavelmente) na capa atual, e continuar o restante em páginas adjacentes.

   Se você publicar ou distribuir cópias Opacas do Documento em número maior que cem (100), você deve necessariamente ou incluir uma cópia Transparente, legível por máquina, junto com cada cópia Opaca, ou declarar, na ou com cada cópia Opaca, uma localização de rede de computador, a partir da qual o público usuário de rede geral tenha acesso para baixar, utilizando protocolos de rede de padrão público, uma cópia Transparente completa do Documento, livre do material adicionado.

   Se você se utilizar da última opção, você deve necessariamente adotar razoavelmente passos prudentes, quando você iniciar a distribuição de cópias Opacas em quantidade, para se assegurar que essa cópia Transparente permanecerá então acessível na localização declarada até pelo menos um ano após a última vez que você distribuiu uma cópia Opaca (diretamente ou por intermédio dos seus agentes ou varejistas) daquela edição ao público.

   É pedido, mas não exigido, que você contate os autores do Documento bem antes de redistribuir qualquer número grande de cópias, para dá-los a oportunidade de lhe fornecer uma versão atualizada do Documento.

4. MODIFICAÇÕES

   Você pode copiar e distribuir uma Versão Modificada do Documento sob as condições das seções dois (2) e três (3) acima, contanto que você publique a Versão Modificada precisamente sob esta Licença, com a Versão Modificada preenchendo a função do Documento, portanto licenciando a distribuição e modificação da Versão Modificada a quem quer que possua uma cópia dela. Adicionalmente, você deve necessariamente fazer estas coisas na Versão Modificada:

   A. Utilize na Página de Título (e nas capas, se existentes) um título distinto daquele do Documento, e daqueles das versões prévias (as quais deveriam, se existiu alguma, serem listadas na seção Histórico do Documento). Você pode utilizar o mesmo título que uma versão prévia, se o editor original daquela versão conceder permissão.

   B. Liste na Página de Título, como autores, uma ou mais pessoas ou entidades responsáveis pela autoria das modificações na Versão Modificada, junto com ao menos cinco dos autores principais do Documento (todos os autores principais, se tiver menos que cinco), a menos que eles liberem você dessa exigência.

C. Declare na Página de Título o nome do editor da Versão Modificada, como o editor.

D. Preserve todos os avisos de direitos autorais do Documento.

E. Adicione um aviso apropriado de direitos autorais para suas modificações, adjacente aos outros avisos de direitos autorais.

F. Inclua, imediatamente após os avisos de direitos autorais, um aviso de licença concedendo ao público permissão para utilizar a Versão Modificada sob os termos desta Licença, na forma mostrada no Adendo abaixo.

G. Preserve, naquele aviso de licença, as listas completas de Seções Invariantes e Textos de Capa exigidos dados no aviso de licença do Documento.

H. Inclua uma cópia inalterada desta Licença.

I. Preserve a seção intitulada "Histórico", Preserve seu Título, e adicione a ele um item declarando ao menos o título, ano, novos autores, e editor da Versão Modificada, conforme dado na Página de Título. Se não existir uma seção intitulada "Histórico" no Documento, crie uma declarando o título, ano, autores, e editor do Documento, conforme dado em sua Página de Título, então adicione um item descrevendo a Versão Modificada, conforme declarado na frase prévia.

J. Preserve a localização de rede, se existente, dada no Documento para acesso público a uma cópia Transparente do Documento, e da mesma forma as localizações de rede dadas no Documento para versões prévias nas quais foi baseado. Essas podem ser colocadas na seção "Histórico". Você pode omitir uma localização de rede para um trabalho que foi publicado nos últimos quatro anos anteriores à publicação do próprio do Documento, ou se o editor original da versão à qual a localização de rede se refere conceder permissão.

K. Para cada seção Intitulada "Agradecimentos" ou "Dedicatórias", Preserve o Título da seção, e preserve na seção toda a substância e tonalidade de cada um dos agradecimentos a contribuidores e/ou dedicatórias dadas nela.

L. Preserve todas as Seções Invariantes do Documento, inalteradas em seus textos e em seus títulos. Os números de Seção ou o equivalente não são considerados parte dos títulos de seção.

M. Delete quaisquer seções Intituladas "Patrocínios". Tal seção não pode ser incluída na Versão Modificada.

N. Não reintitule qualquer seção existente para Intitulada "Patrocínios" ou para conflitar no título com qualquer Seção Invariante.

O. Preserve quaisquer Declarações de Garantia.

Se a Versão Modificada incluir novas seções pré textuais ou apêndices que se qualifiquem como Seções Secundárias e não contenham material copiado a partir do Documento, você pode, a sua escolha, designar algumas ou todas essas seções como Invariantes. Para fazer isso, adicione seus títulos à lista das Seções Invariantes no aviso de licença da Versão Modificada. Esses títulos devem necessariamente serem distintos de quaisquer outros títulos de seções.

Você pode adicionar uma seção Intitulada "Patrocínios", contanto que ela não contenha nada além de patrocínios da sua Versão Modificada por vários patrocinadores—por exemplo, declarações de avaliadores ou aquelas de que o texto foi aprovado por uma organização como a definição autorizativa de um padrão.

Você pode adicionar uma passagem de até cinco palavras, como um Texto de Primeira Capa, e uma passagem de até vinte e cinco palavras, como um Texto de Quarta Capa, ao final da lista dos Textos de Capa na Versão Modificada. Somente uma passagem de Texto de Primeira Capa e uma de Texto de Quarta Capa podem ser adicionadas por (ou mediante

acordos feitos por) qualquer uma entidade. Se o Documento já inclui um texto de capa para a mesma capa, previamente adicionado por você ou por acordo feito pela mesma entidade pela qual você está atuando, você não pode adicionar outro; porém você pode substituir o antigo, na permissão explícita do editor prévio que adicionou o antigo.

O(s) autor(s) e editor(s) do Documento, por esta Licença, não concedem permissão para utilizar seus nomes para publicidade para ou para afirmar ou implicar patrocínio de qualquer Versão Modificada.

5. COMBINANDO DOCUMENTOS

   Você pode combinar o Documento com outros documentos publicados sob esta Licença, sob os termos definidos na seção quatro (4) acima para versões modificadas, contanto que você inclua na combinação todas as Seções Invariantes de todos os documentos originais, não modificados, e listá-los todos como Seções Invariantes do seu trabalho combinado no seu aviso de licença, e você preserva todas as Declarações de Garantias deles.

   O trabalho combinado precisa conter somente uma cópia desta Licença, e múltiplas Seções Invariantes idênticas podem ser substituídas por uma cópia única. Se existirem múltiplas Seções Invariantes com o mesmo nome, mas conteúdos diferentes, torne o título de cada uma de tal seção único adicionando ao final dele, entre parênteses, o nome do autor ou editor original daquela seção se conhecido, ou, do contrário, um número único. Faça o mesmo ajuste aos títulos da seção na lista de Seções Invariantes no aviso de licença do trabalho combinado.

   Na combinação, você deve necessariamente combinar quaisquer seções Intituladas "Histórico" nos vários documentos originais, formando uma seção Intitulada "Histórico"; de mesma maneira, combine quaisquer seções Intituladas "Agradecimentos", e quaisquer seções Intituladas "Dedicatórias". Você deve necessariamente deletar todas as seções Intituladas "Patrocínios".

6. COLEÇÕES DE DOCUMENTOS

   Você pode produzir uma coleção consistente do Documento e outros documentos publicados sob esta Licença, e substitua as cópias individuais desta Licença nos vários documentos por uma cópia única que esteja incluída na coleção, contanto que você siga as regras desta Licença para cópias literais de cada um dos documentos em todos os outros aspectos.

   Você pode extrair um documento único de tal coleção, e distribuí-lo individualmente sob esta Licença, contanto que você insira uma cópia desta Licença no documento extraído, e siga esta Licença em todos os outros aspectos relativos à cópias literais daquele documento.

7. AGREGAÇÃO COM TRABALHOS INDEPENDENTES

   Uma compilação do Documento ou seus derivados com outros documentos separados e independentes ou trabalhos, dentro ou junto a volume de armazenamento ou meio de distribuição, é chamado em "agregado" se os direitos autorais resultantes da compilação não forem utilizados para limitar os direitos legais dos usuários da compilação além do que os trabalhos individuais permitem. Quando o Documento for incluído em um agregado, esta Licença não se aplica aos outros trabalhos no agregado, os quais não são eles próprios trabalhos derivados do Documento.

   Se a exigência do Texto de Capa da seção três (3) for aplicável a essas cópias do Documento, então se o Documento for menor que a metade do agregado inteiro, os Textos de Capa do Documento podem ser colocados em capas que encartem o Documento dentro do agregado, ou o equivalente eletrônico de capas se o Documento estiver em formato eletrônico. Do contrário, eles devem necessariamente aparecer nas capas impressas que encartem o agregado inteiro.

8. TRADUÇÃO

   Tradução é considerada um tipo de modificação, de forma que você pode distribuir traduções do Documento sob os termos da seção quatro (4). A substituição de Seções Invariantes por

traduções exige permissão especial de seus detentores dos direitos autorais, porém você pode incluir traduções de algumas ou todas as Seções Invariantes adicionalmente às versões originais dessas Seções Invariantes. Você pode incluir uma tradução desta Licença, e todos os avisos de licença no Documento, e quaisquer Declarações de Garantia, contanto que você inclua também a versão original em Inglês desta Licença e as versões originais daqueles avisos e declarações. No caso de uma divergência entre a tradução e a versão original desta Licença ou um aviso ou declaração, a versão original prevalecerá.

Se uma seção no Documento for Intitulada "Agradecimentos", "Dedicatórias", ou "Histórico", a exigência (seção 4) de Preservar seu Título (seção 1) tipicamente exigirá a modificação do título atual.

9. FINALIZAÇÃO

   Você não pode copiar, modificar, sublicenciar, ou distribuir o Documento, exceto conforme expressamente provido sob esta Licença. Qualquer tentativa clandestina de copiar, modificar, sublicenciar, ou distribuir o Documento é inválida, e automaticamente finalizará seus direitos sob esta Licença.

   Entretanto, se você cessar todas as violações a esta Licença, então a sua licença oriunda de um detentor de direitos autorais em particular está restabelecida (a) provisoriamente, a menos e até que o detentor dos direitos autorais explicita e finalmente cancele sua licença; e (b) permanentemente, se o detentor dos direitos autorais falhar em notificar você da violação, por algum meio razoável, antes de sessenta (60) dias após a cessação.

   Além disso, a sua licença oriunda de um detentor de direitos autorais em particular está restabelecida permanentemente se o detentor dos direitos autorais notificar você sobre a violação por algum meio razoável, essa for a primeira vez que você recebeu um aviso de violação desta Licença (para qualquer trabalho) oriunda daquele detentor de direitos autorais, e você sanar a violação antes de decorridos trinta (30) dias após o seu recebimento do aviso.

   A finalização dos seus direitos sob esta seção não finaliza as licenças de varejistas que tenham recebido cópias ou direitos de você sob esta Licença. Se os seus direitos tiverem sido finalizados e não permanentemente restabelecidos, o recebimento de uma cópia de algum ou de tudo do mesmo material não concede a você direitos de utilizá-lo.

10. REVISÕES FUTURAS DESTA LICENÇA

    A Free Software Foundation pode publicar novas, revisadas versões da Licença de Documentação Livre GNU de tempos em tempos. Tais novas versões serão similares na essência à presente versão, porém podem diferir em detalhes para abarcar novos problemas ou assuntos. Veja-se `http://www.gnu.org/copyleft/`.

    Para cada versão da Licença é dado um número distintivo de versão. Se o Documento especifica que uma versão numerada em particular desta Licença "ou qualquer versão posterior" se aplica a ele, você tem a opção de seguir os termos e condições ou da versão especificada ou de qualquer versão posterior que tenha sido publicada (não como um rascunho) pela Free Software Foundation. Se o Documento não especifica um número de versão desta Licença, você pode escolher qualquer versão já publicada (não como um rascunho) pela Free Software Foundation. Se o Documento especifica que um procurador pode decidir quais versões futuras desta Licença podem ser utilizadas, essa declaração pública do procurador de aceitação de uma versão permanentemente autoriza você a escolher aquela versão para o Documento.

11. RELICENCIAMENTO

    "Sítio de Colaboração Massiva Multi autor" (ou "Sítio MMC") significa qualquer servidor da Rede Mundial de Computadores que publica trabalhos sujeitos a direitos autorais e também provê facilidades proeminentes para qualquer pessoa editar esses trabalhos. Um wiki público que qualquer pessoa pode editar é um exemplo de tal servidor. Uma "Colaboração Massiva

Multi autor" (ou "MMC") contida no sítio significa qualquer conjunto de trabalhos sujeitos a direitos autorais assim publicados no sítio MMC.

"CC-BY-SA" significa a licença Creative Commons Attribution-Share Alike 3.0 publicada pela Creative Commons Corporation, uma corporação sem fins lucrativos com seu domicílio empresarial situado em São Francisco, Califórnia, Estados Unidos da América do Norte, bem como versões futuras de esquerdos autorais dessa licença publicadas pela mesma organização.

"Incorporar" significa publicar ou republicar um Documento, no todo ou em parte, como parte de outro Documento.

Um MMC é "elegível para relicenciamento" se ele for licenciado sob esta Licença, e se todos os trabalhos que foram primeiro publicados sob esta Licença em algum lugar que não esse MMC, e subsequentemente incorporados, no todo ou em parte, no MMC, (1) não tinham textos de capa ou seções invariantes; e (2) estavam assim incorporados antes de 01 de novembro de 2008.

O operador de um Sítio MMC pode republicar um MMC contido no sítio sob CC-BY-SA, no mesmo sítio, a qualquer tempo antes de 01 de agosto de 2009, contanto que o MMC seja elegível para relicenciamento.

ADENDO: Como utilizar esta Licença para seus documentos

Para utilizar esta Licença em um documento que você escreveu, inclua um cópia da Licença no documento e coloque os seguintes avisos de direitos autorais e licença pouco depois da página de título:

```
Direitos autorais (C) ano  seu nome.
Permissão é concedida para copiar, distribuir e/ou modificar este
documento sob os termos da Licença de Documentação Livre GNU, Versão
1.3 ou qualquer versão posterior publicada pela Free Software
Foundation; sem Seções Invariantes, sem Textos de Primeira Capa, e sem
Textos de Quarta Capa.   Uma cópia da licença está inclusa na seção
intitulada ``Licença de Documentação Livre GNU''.
```

Se você tiver Seções Invariantes, Textos de Primeira Capa e Textos de Quarta Capa, substitua a linha "sem. . . Capa" por isto:

```
com as Seções Invariantes sendo liste seus títulos, com os
Textos de Primeira Capa sendo lista, e com os Textos de Quarta
Capa sendo lista.
```

Se você tiver Seções Invariantes sem Textos de Capa, ou alguma outra combinação dos três, mescle essas duas alternativas para adequar a situação.

Se o seu documento contém exemplos não triviais de código de programação, nós recomendamos publicar esses exemplos em paralelo, sob sua escolha de licença de software livre, tal como a Licença Pública Geral GNU, para permitir seu uso em software livre.

Índice de Comando e Variável

Esta é uma lista alfabética de todos os comandos @, funções Emacs Lisp variadas e diversas variáveis. Para tornar a lista mais fácil de usar, os comandos estão listados sem o '`@`' precedente.

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Índice Geral

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z