# Texinfo

# Texinfo

Формат документации GNU
для Texinfo версии 4.0, 28 сентября 1999г

**Роберт Дж. Чассел**
**Ричард М. Столмен**

Это руководство относится к Texinfo версии 4.0, 28 сентября 1999г.



Рисунок на обложке Этьена Сюваса.

# Краткое содержание

# Оглавление

Documentation is like sex: when it is good, it is very, very good; and when it is bad, it is better than nothing. —Dick Brandon

# Условия копирования Texinfo

Распространяемые на данный момент программы, имеющие отношение к Texinfo, включают фрагменты GNU Emacs, а также отдельные программы (в том числе `makeinfo`, `info`, `texindex` и '`texinfo.tex`'). Эти программы *свободны*; это значит, что каждый может использовать и распространять на свободной основе. Программы, имеющие отношение к Texinfo, не являются общественным достоянием; на них распространяется авторское право, и на их распространение существуют ограничения, но эти ограничения разработаны так, чтобы позволить все, что хороший благонамеренный гражданин может захотеть. Что не разрешается, так это пытаться мешать другим и далее совместно пользоваться любой версией этих программ, которые они могли бы от вас получить.

А именно, мы хотим убедиться, что у вас есть право распространять копии программ, имеющих отношение к Texinfo, что вы получаете исходный код или можете получить его, если захотите, что вы можете изменять эти программы или использовать их части в новых свободных программах и что вы знаете, что вы можете все это делать.

Чтобы убедиться, что каждый имеет такие права, мы должны запретить вам лишать кого-либо этих прав. Например, если вы распространяете копии программ, имеющих отношение к Texinfo, вы должны предоставить получателям все права, которыми обладаете вы сами. Вы обязаны убедиться, что они тоже получат или смогут получить исходный код. И вы должны сообщить им об их правах.

Также, для нашей собственной защиты, мы хотим удостовериться, что все понимают, что гарантий на программы, имеющие отношение к Texinfo, нет. Если эти программы модифицируются и передается кем-то еще, мы хотим, чтобы получатели знали, что то, что у них есть — это не то, что распространяем мы, чтобы любые проблемы, созданные другими, не отразились на нашей репутации.

Точные условия лицензий для распространяемых на данный момент программ, имеющих отношение к Texinfo, находятся в поставляемой с ними Универсальной Общественной Лицензии GNU.

# 1 Обзор Texinfo

*Texinfo*[1] — это система создания документации, которая использует единый входной файл для производства как интерактивной информации, так и распечатки. Это означает, что вместо написания двух разных документов, одного для интерактивной информации, а другого для печатного произведения, вам нужно написать только один. Следовательно, при внесении изменений вам нужно будет отредактировать только один документ.

## 1.1 Описание ошибок

Мы рады получить ваши предложения по улучшению системы Texinfo и сообщения об ошибках как для программ, так и для документации. Пожалуйста, посылайте их по адресу `bug-texinfo@gnu.org`. Вы можете получить последнюю версию Texinfo на `ftp://ftp.gnu.org/gnu/texinfo/` и с зеркал по всему миру.

При описании ошибок, пожалуйста, включайте достаточно информации, чтобы сопроводители могли воспроизвести проблему. Вообще говоря, это означает:

- номер версии Texinfo и рассматриваемой программы или руководства,
- версии оборудования, операционной системы и компилятора,
- любые необычные параметры, которые вы задали при конфигурировании,
- содержимое всех входных файлов, необходимых для воспроизведения проблемы,
- описание проблем и примеры любого ошибочного вывода,
- все остальное, что, по вашему мнению, может помочь.

Если вы сомневаетесь, необходимы ли какие-то сведения или нет, включайте их. Лучше включить слишком много, чем пропустить что-то важное.

Особо приветствуются готовые исправления; если возможно, пожалуйста, делайте их с помощью ‘`diff -c`’ (см. раздел “Overview” в *Comparing and Merging Files*) и включайте описание ваших изменений для журнала ‘`ChangeLog`’ (см. раздел “Change Log” в *Руководство по GNU Emacs*).

При посылке сообщения, если возможно, не кодируйте и не разбивайте его; гораздо проще обращаться с простым текстовым сообщением, даже большим, чем с многими маленькими. Удобный способ упаковки нескольких и/или двоичных файлов для электронной почты предоставляет GNU shar.

## 1.2 Использование Texinfo

Используя Texinfo, вы можете создавать печатный документ с обычными для книг свойствами, включая главы, разделы, перекрестные ссылки и именные указатели. Из этого же исходного Texinfo-файла вы можете создать интерактивный, управляемый

[1] Первый слог в слове “Texinfo” произносится как “тек”, а не “текс”. Такое необычное произношение происходит от произношения слова TEX, но не идентично ему. Буква ‘`X`’ в слове TEX, на самом деле, — это греческая буква “хи”, а не английская “экс”. Произносите TEX, как если бы ‘`X`’ был последним звуком имени ‘Bach’; но в Texinfo произносите ‘`x`’ как ‘к’. Пишите “Texinfo” с заглавной “T” и остальными строчными буквами.

меню Info-файл с нодами, меню, перекрестными ссылками и именными указателями. Кроме того, из этого же исхдного файла вы можете создать HTML-файл для просмотра с помощью Web-броузера. Хорошими примерами Texinfo-файлов служат *Руководство по GNU Emacs* и данное руководство.

Чтобы получить печатный документ, вы пропускаете исходный Texinfo-файл через форматирующую программу TEX (но язык Texinfo сильно отличается от plain TEX, обычного языка TEX). Она создаст DVI-файл, который вы сможете распечатать в виде книги или отчета (см. Глава 19 [Печать], с. 146).

Для получения Info-файла вам нужно обработать Texinfo-файл с помощью программы `makeinfo` или команды Emacs `texinfo-format-buffer`. Вы сможете установить созданный Info-файл в дерево системы Info, (см. Раздел 20.2 [Установка Info-файла], с. 167).

Чтобы сделать HTML-файл, обработайте ваш исходный Texinfo-файл с помощью `makeinfo`, используя ключ '`-html`'. Вы можете (к примеру) установить результат на вашем Web-сайте.

Если вы программист и хотели бы внести свой вклад в проект GNU, реализовав дополнительные выходные форматы для Texinfo, это было бы великолепно. Но, пожалуйста, не пишите отдельный транслятор texi2foo для вашего любимого формата foo! Это трудный путь выполнения этой задачи, он создает дополнительную работу для дальнейшего сопровождения, так как язык Texinfo постоянно расширяется и обновляется. Вместо этого, лучшим вариатом будет изменить `makeinfo` так, чтобы она генерировала вывод в новом формате, как сейчас она делает для Info и HTML.

TEX работает практически со всеми принтерами; а Info — почти с любым типом терминалов; HTML-вывод работает практически со всеми web-броузерами. Поэтому практически все пользователи компьютернов могут применять Texinfo.

Исходный Texinfo-файл — это простой ASCII-файл, содержащий текст и `@`-*команды* (слова, начинающиеся с '`@`'), сообщающие форматирующим и подготавливающим к печати программам, что нужно делать. Вы можете редактировать Texinfo-файл любым текстовым редактором; но особенно удобно будет использовать GNU Emacs, так как он имеет специальный режим, называемый режимом Texinfo, предоставляющий многие возможности специально для Texinfo. (См. Глава 2 [Режим Texinfo], с. 15.)

Перед написанием Texinfo-файлов вам стоит ознакомиться с тем, что такое ноды, меню, перекрестные ссылки и прочее, например, читая данное руководство.

Вы можете использовать Texinfo для создания как интерактивной информации, так и печатных руководств, кроме того, Texinfo распространяется свободно. Поэтому Texinfo — это официальный формат документации для утилит и библиотек проекта GNU. Подробная информация доступна на странице документации GNU.

Время от времени выдвигаются предложения генерировать традиционные страницы man Unix из исходников на Texinfo. Скорее всего, это никогда не будет поддерживаться, поскольку страницы man имеют очень строгий условный формат. Простого расширения `makeinfo` для поддержки формата troff было бы недостаточно. Создание хорошей страницы man, следовательно, требует совершенно иного исходного текста, в отличие от типичного применения Texinfo — создания хорошего руководства пользователя и справочного руководства. Поэтому создание страниц man не совмещается с

целью Texinfo — не документировать одну и ту же информацию несколькими способами для разных выходных форматов. Вы также можете просто написать страницу man саму по себе.

Если вы хотите поддерживать страницы man, может оказаться полезной программа `help2man`; она генерирует традиционную страницу man из вывода программы по ключу '`-help`'. Фактически, в настоящее время она применяется для генерации страниц man для самих программ Texinfo. Это свободная программа, написанная Бренданом О'Ди и доступная на http://www.ozemail.com.au/~bod/help2man.tar.gz.

## 1.3 Info-файлы

Info-файл — это Texinfo-файл, отформатированный таким образом, чтобы его можно было использовать с программой просмотра документации Info. (`makeinfo` и `texinfo-format-buffer` — две команды, преобразующие Texinfo-файл в Info-файл.)

Info-файлы разделены на секции, называемые *нодами*, в каждой из которых рассмотрена одна тема. Каждая нода имеет имя и содержит как текст, предназначенный для чтения пользователем, так и указатели на другие ноды, различаемые по именам. Программа Info отображает в каждый момент одну ноду и предоставляет команды, с помощью которых пользователь может передвигаться к другим родственным нодам.

Каждая нода Info может содержать любое число дочерних нод, описывающих подтемы рассматриваемой в ноде темы. Имена дочерних нод перечислены в *меню* родительской ноды; это позволяет передвигаться к одной из дочерних нод, используя определенные команды Info. Обычно Info-файл организован подобно книге. Если нода находится на логическом уровне главы, то дочерние ноды находятся на уровне разделов; дочерние ноды разделов находятся, соответственно, на уровне подразделов.

Все дочерние и родительские ноды связаны в двунаправленную цепочку указателей 'Next' и 'Previous' ('Следующая' и 'Предыдущая'). Указатель 'Next' предоставляет связь со следующим разделом, а 'Previous' — с предыдущим. Это означает, что все ноды на уровне разделов одной главы связаны вместе. Как правило, порядок нод в этой цепочке такой же, как и порядок дочерних нод в родительском меню. Каждая дочерняя нода хранит указатель на родительскую ноду под именем 'Up' ('Вверх'). Последний потомок не имеет указателя 'Next', а первый потомок указывает на родителя и в качестве 'Previous', и 'Up'.[2]

Организация Info-файла по типу книги, с главами и разделами, — это только вопрос соглашения, а не необходимое условие. Указатели 'Up', 'Previous' и 'Next' могут указывать на любые ноды, а меню также может содержать любые другие ноды. Таким образом, структура нод может быть произвольным направленным графом. Но обычно для понятности лучше следовать структуре глав и разделов печатной книги или отчета.

Кроме меню и указателей 'Next', 'Previous' и 'Up' Info предоставляет указатели другого рода, называемые ссылками, которые могут появляться в любом месте текста. Как правило, это лучший способ представлять ссылки, не попадающие в иерархическую структуру.

[2] В некоторых документах первый потомок не имеет указателя 'Previous'. Иногда последний потомок указывает как на 'Next' на следующую ноду более высокого уровня.

Обычно вы разрабатываете документ так, чтобы ноды соответствовали структуре глав и разделов печатной версии. Но иногда это не подходит для обсуждаемого материала. Тогда Texinfo использует отдельные команды для задания структуры нод и структуры разделов печатной версии.

Обычно вы начинаете просмотр Info-файла с ноды, называемой по соглашению ‘Top’. Эта нода содержит лишь краткий обзор целей файла и большое меню, через которое можно получить доступ к остальной части файла. Из этой ноды вы можете просмотреть файл последовательно ноду за нодой или перейти к конкретной ноде, перечисленной в главном меню, или найти в именных указателях непосредственно ту ноду, которая содержит нужную вам информацию. Кроме того, при использовании самостоятельной программы Info вы можете задать определенные пункты меню в командной строке (см. раздел “Top” в *Info*).

Если вы хотите прочитать Info-файл последовательно, как печатное руководство, вы можете постоянно нажимать ⟨SPC⟩ или получить Info-файл целиком продвинутой командой Info `g *`. (См. Info файл ‘`info`’, node ‘`Expert`’.)

Файл ‘`dir`’ в каталоге ‘`info`’ служит отправной точкой для системы Info в целом. Из него вы можете перейти к первым (‘Top’) нодам каждого документа во всей системе Info.

Если вы хотите сослаться на Info-файл в URI, вы можете использовать (неофициальный) синтаксис, пример которого приведен ниже. Это работает только с Emacs/W3, к примеру:

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

Сама программа `info` не следует по URI любого вида.

## 1.4 Печатные книги

Texinfo-файл может быть отформатирован и набран для печати книги или руководства. Чтобы сделать это, вам понадобится TEX, мощная и сложная программа компьютерного набора, написанная Дональдом Кнутом.[3]

Книга, основанная на Texinfo, подобна любой другой набранной печатной книге: у нее может быть титульный лист, страница с информацией об авторских правах, содержание и предисловие, а также главы, нумерованные или ненумерованные разделы и подразделы, заголовки страниц, перекрестные ссылки, сноски и именные указатели.

Вы можете использовать Texinfo для написания книги, даже не замышляя преобразовать ее в интерактивный вид. Вы можете использовать Texinfo для написания повести или даже печатного меморандума, хотя последние не рекомендуется, так как для этого намного лучше подходит электронная почта.

TEX — программа верстки общего назначения. С Texinfo поставляется файл ‘`texinfo.tex`’, который содержит информацию (определения или *макросы*), которую

[3] Вы также можете использовать программу `texi2roff`, если у вас нет TEX; так как Texinfo предназначена для использования с TEX, `texi2roff` здесь не рассматривается. `texi2roff` не входит в стандартную поставку GNU, не поддерживается и не соответствует всем последним возможностям языка Texinfo, описанным в данном руководстве.

TEX использует при наборе Texinfo-файла. (‘`texinfo.tex`’ сообщает TEX как переводить `@`-команды Texinfo в команды TEX, которые он уже может обработать для создания набранного документа.) ‘`texinfo.tex`’ содержит спецификации для печати документа. Вы можете получить последнюю версию файла ‘`texinfo.tex`’ по адресу `ftp://ftp.gnu.org/gnu/texinfo.tex`.

Чаще всего документы печатают на страницах размером 8.5 на 11 дюймов (216 мм на 280 мм, это размер по умолчанию), но вы также можете печатать на страницах размером 7 на 9.25 дюймов (178 мм на 235 мм, размер `@smallbook`) или на европейском формате A4 (`@afourpaper`). (См. Раздел 19.11 [Печать “маленьких” книг], с. 154, а также Раздел 19.12 [Печать на формате A4], с. 155.)

Вы можете изменить размер печатной версии, изменяя параметры в файле ‘`texinfo.tex`’. Помимо размера, вы можете изменить стиль форматирования; например, изменить размеры шрифтов и сами используемые шрифты, размеры отступов абзацев, степень разбиения слов при переносах и другие параметры. Изменяя установки, вы можете сделать книгу величественной, старомодной и серьезной, или легкомысленной, молодой и веселой.

TEX распространяется свободно. Он написан на языке WEB, расширении языка Паскаль, и может быть скомпилирован из текстов на Паскале или Си (с применением программы перевода, поставляемой вместе с TEX). (См. раздел “Режим TEX” в *Руководство по GNU Emacs*, для получения информации о TEX.)

TEX — очень мощная программа и имеет много возможностей. Но ввиду того, что Texinfo-файл должен предоставлять информацию и на текстовых терминалах, и в виде книги, набор форматирующих команд, поддерживаемых Texinfo, был по необходимости ограничен.

См. Приложение J [Как получить TEX], с. 221.

## 1.5 `@`-команды

Команды Texinfo, сообщающие TEX, каким образом нужно набирать печатное руководство, а `makeinfo` и `texinfo-format-buffer` — как создать Info-файл, начинаются с символа ‘`@`’; они называются *`@`-командами*. Например, `@node` — это команда, указывающая начало ноды, а `@chapter` — указывающая начало главы.

> **Обратите внимание:** Все `@`-команды за исключением `@TeX{}` должны набираться строчными буквами.

`@`-команды Texinfo составляют строго ограниченный набор конструкций. Строгое ограничение позволяет понимать Texinfo-файлы как TEX, так и командам, создающим Info-файлы. Вы можете просмотреть Info-файлы на любом алфавитно-цифровом терминале и распечатать вывод TEX на многих типах принтеров.

Вы должны писать `@`-команды в отдельной строке или внутри предложения в зависимости от их действия и принимаемых аргументов[4]:

[4] Слово *аргумент* употребляется здесь в смысле, принятом в математике, и не имеет отношения к дискуссиям между людьми; оно относится к информации, передаваемой команде. В соответствии со словарем *Oxford English Dictionary*, это слово происходит от латинского *разъяснять, доказывать*; таким образом, оно означает ‘очевидность, предоставленная как доказанное’, то есть ‘предоставленная информация’, что приводит к его значению в математике. В другой ветви

- Пишите команды вроде `@noindent` с начала отдельной строки. (`@noindent` отменяет отступ для начала абзаца в следующей строке.)
- Пишите команды вроде `@chapter` с начала строки с последующими аргументами, в данном случае названием главы, в конце строки. (`@chapter` создает названия глав.)
- Пишите команды вроде `@dots{}` где вы хотите, обычно внутри предложения. (`@dots{}` вставляет многоточие . . .)
- Пишите команды вроде `@code{`*код*`}`, где вы хотите (но обычно внутри предложения) с аргументом, в данном случае *код*, заключенным в фигурные скобки. (`@code` отмечает программный код.)
- Пишите команды вроде `@example` на отдельной строке; текст тела пишите на последующих строках; пишите завершающую команду `@end`, в данном случае `@end example`, на отдельной строке после текста тела. (`@example ... @end example` делает отступы и набирает текст тела в виде примера.) Обычно определяющие среду команды, подобные этой, можно писать с отступом, но в сложных и трудноопределяемых обстоятельствах дополнительные промедутки на входе производят лишние промежутки на выводе, поэтому будьте внимательны.

Как правило, команды, встречающиеся внутри текста, нужно использовать с фигурными скобками, но команды, записываемые на отдельной строке, не нужно. Исключение составляют неалфавитные команды, такие как `@:`: они не требуют скобок.

По мере появления у вас опыта в использовании Texinfo, вы быстро научитесь, как писать разные команды: различные способы написания команд позволяют легче писать и читать Texinfo-файлы, чем если бы все команды следовали единому синтаксису. (Подробности о синтаксисе @-команд смотрите в Приложение I [Синтаксис @-команд], с. 220.)

## 1.6 Основные соглашения о синтаксисе

Этот раздел описывает общие соглашения, применяемые во всех документах Texinfo.

- В Texinfo-файле могут появиться в своих обычных значениях все печатные ASCII-символы, кроме ‘`@`’, ‘`{`’ и ‘`}`’. ‘`@`’ — это сигнальный символ, с которого начинаются все команды. ‘`{`’ и ‘`}`’ могут быть использованы только для группирования аргументов некоторых команд. Чтобы вставить эти специальные символы в ваш документ, поместите перед ними символ ‘`@`’, например: ‘`@@`’, ‘`@{`’ и ‘`@}`’.
- В TEX обычно используются два символа одинарных кавычек для заключения текста в двойные кавычки: ■ и ■ . Это соглашение должно соблюдаться и в Texinfo-файлах. TEX преобразует символы одинарных кавычек в правые и левые двойные кавычки, а Info — в символ `"` , двойные кавычки ASCII.
- Используйте три дефиса подряд, ‘`---`’, для получения такого — тире. В TEX один или два дефиса дают при печати тире, которое короче обычного печатного тире. Info для отображения на экране сокращает три дефиса до двух.

---

заимствования это слово стало означать ‘высказываться так, чтобы другие могли высказывать противоположные суждения’, что привело к значению слова ‘аргумент’ относительно к спору.

- Для предотвращения в печатном руководстве отступа как для начала абзаца, поместите перед абзацем команду `@noindent` на отдельной строке.
- Если вы отметите область Texinfo-файла командами `@iftex` и `@end iftex`, то она появится только в печатной версии; в этой области вы можете использовать команды из plain TEX, которые недоступны в Info. Аналогично, если вы отметите область командами `@ifinfo` и `@end ifinfo`, то она появится только в Info-файле; в этой области вы сможете использовать команды Info, недоступные в TEX. То же и для `@ifhtml ... @end ifhtml`, `@ifnothtml ... @end ifnothtml`, `@ifnotinfo ... @end ifnotinfo`, `@ifnottex ... @end ifnottex`. См. Глава 16 [Условия], с. 134.

  **Внимание:** Не используйте символы табуляции в Texinfo-файлах! В TEX применяются шрифты переменной ширины, а это значит, что вы не сможете обеспечить правильное действие табуляции в каждом случае. Поэтому TEX воспринимает символы табуляции как один пробел, а это не похоже на табуляцию. Кроме того, `makeinfo` не предпринимает никаких особых действий для символов табуляции, таким образом, вставляя их на вход, вы можете получить что-то, чего не ожидали.

  Чтобы избежать этой проблемы, режим Texinfo заставляет GNU Emacs вставлять несколько пробелов, когда вы нажимаете клавишу ⟨TAB⟩.

  Вы также можете запустить в Emacs функцию `untabify`, чтобы превратить в области все символы табуляции в пробелы.

## 1.7 Комментарии

Вы можете писать в Texinfo-файле комментарии, которые не появятся ни в Info-файле, ни в печатном руководстве, используя команду `@comment` (ее можно сокращать как `@c`). Такие комментарии предназначены для человека, пересматривающего Texinfo-файл. Весь текст в строке после `@comment` или `@c` является комментарием. (Чаще всего вы можете написать `@comment` или `@c` в середине строки, и только текст, следующий за этими командами не появится в результате; но некоторые команды, например `@settitle` и `@setfilename` действуют на всю строку. Вы не можете использовать `@comment` или `@c` в строке, начинающейся с такой команды.)

Вы можете написать большие куски текста, которые не появятся ни в Info-файле, ни в печатном руководстве, используя команды `@ignore` и `@end ignore`. Пишите каждую из этих команд с начала отдельной строки. Текст, заключенный между двумя этими командами, не появляется при выводе. Таким образом, вы можете использовать `@ignore` и `@end ignore` для написания комментариев. Часто `@ignore` и `@end ignore` применяют, чтобы ограничить фрагмент разрешения на копирование, относящийся к исходному тексту Texinfo, но не к Info-файлу или печатному руководству.

## 1.8 Что должен содержать Texinfo-файл

По соглашению, имена Texinfo-файлов заканчиваются расширением ‘`.texinfo`’, ‘`.texi`’, ‘`.txi`’ или ‘`.tex`’. Предпочтительны длинные расширения, так как они яснее показывают читающему природу файла. Короткие расширения нужны для систем, которые не умеют обращаться с длинными именами файлов.

Чтобы получился Info-файл или печатное руководство, Texinfo-файл **должен** начинаться такими строками:

```
\input texinfo
@setfilename имя-info-файла
@settitle название-руководства
```

За этим началом следует содержимое файла, и вы **должны** завершить Texinfo-файл такой строкой:

```
@bye
```

Строка '`\input texinfo`' велит TEX загрузить файл '`texinfo.tex`', который сообщает TEX, как переводить `@`-команды Texinfo в команды набора TEX. (Обратите внимание на использование обратной косой черты, '\'; это является правильным для TEX.) Строка '`@setfilename`' задает имя Info-файла и велит TEX открывать вспомогательные файлы. Строка '`@settitle`' задает название печатного руководства для заголовков страниц.

Строка `@bye` в конце файла сообщает форматирующим программам, что файл пройден целиком и нужно остановить форматирование.

Обычно вы будете использовать не этот, довольно скудный формат, а будете включать в начало Texinfo-файла строки start-of-header и end-of-header (начало-заголовка и конец-заголовка), например так:

```
\input texinfo   @c -*- texinfo -*-
@c %**start of header
@setfilename имя-info-файла
@settitle название-руководства
@c %**end of header
```

'`-*- texinfo -*-`' в первой строке заставляет Emacs переключиться в режим Texinfo, когда вы хотите редактировать файл.

Строки `@c`, окружающие строки '`@setfilename`' и '`@settitle`', не обязательны, но понадобятся, если вы захотите запустить TEX или Info только для части файла. (См. Раздел 3.2.2 [Начало заголовка], с. 30, для большей информации.)

Кроме того, обычно вы будете снабжать Texinfo-файл титульным листом, именными указателями и другими элементами. Но необходимый минимум, который может понадобиться для коротких документов, — это лишь три строки в начале и одна в конце.

## 1.9 Шесть частей Texinfo-файла

Вообще говоря, Texinfo-файл содержит что-нибудь еще кроме минимально необходимых начала и конца — обычно он состоит из шести частей:

1. Заголовок

   *Заголовок* задает имя файла, сообщает TEX, какой форматный файл использовать, и выполняет другие служебные задачи.

2. Обзорное описание и авторские права

   Сегмент *Обзорное описание и авторские права* описывает документ и содержит информацию об авторских правах и разрешениях на копирование

для Info-файла. Этот сегмент должен быть заключен между командами `@ifinfo` и `@end ifinfo`, чтобы форматирующие программы помещали его только в Info-файл.

3. Название и авторские права
   Сегмент *Название и авторские права* содержит название и информацию об авторских правах и разрешениях на копирование для печатного руководства. Этот сегмент должен быть заключен между командами `@titlepage` и `@end titlepage`. Название и страница с информацией об авторских правах появляется только в печатном руководстве.

4. Нода ‘Top’ и главное меню
   *Главное меню* содержит полное меню со всеми нодами Info-файла. Оно появляется только в Info-файле в первой ноде (ноде ‘Top’).

5. Тело
   *Тело* документа может быть структурировано подобно традиционной книге или энциклопедии или в свободной форме.

6. Конец
   В *Конце* содержатся команды для печати именных указателей и составления содержания, а также команда `@bye` на отдельной строке.

## 1.10 Короткий пример Texinfo-файла

Вот завершенный, но очень короткий Texinfo-файл из шести частей. Первые три части файла, от ‘`\input texinfo`’ до ‘`@end titlepage`’, кажутся более запугивающими, чем они есть на самом деле. Большая часть материала — это стандартный шаблон; когда вы пишите руководство, просто вставьте в этот сегмент названия для него. (См. Глава 3 [Начало файла], с. 28.)

Текст примера далее содержит *omcmyn*; комментарии не содержат. Полный файл, без комментариев, показан в Приложение C [Пример Texinfo-файла], с. 197.

### Часть 1: Заголовок

Заголовок не появляется ни в Info-файле, ни в печатном документе. В нем устанавливаются различные параметры, включая имя Info-файла и название, используемое в заголовках печатной версии.

```
\input texinfo   @c -*-texinfo-*-
@c %**start of header
@setfilename sample.info
@settitle Пример документа
@setchapternewpage odd
@c %**end of header
```

### Часть 2: Обзорное описание и авторские права

Сегмент с обзорным описанием и авторскими правами не появляется в печатном документе.

```
@ifinfo
Это короткий пример законченного Texinfo-файла.

Copyright @copyright{} 1990 Free Software Foundation, Inc.
@end ifinfo
```

### Часть 3: Титульный лист и авторские права

Сегмент с титульным листом и авторскими правами не появляется в Info-файле.

```
@titlepage
@sp 10
@comment Заголовок печатается крупным шрифтом
@center @titlefont{Пример заголовка}

@c Две следующие команды начинают страницу с информацией об
@c авторских правах
@page
@vskip 0pt plus 1filll
Copyright @copyright{} 1990 Free Software Foundation, Inc.
@end titlepage
```

### Часть 4: Нода ‘Top’ и главное меню

Нода ‘Top’ содержит главное меню Info-файла. Так как в печатном руководстве вместо меню используется содержание, главное меню появляется только в Info-файле.

```
@node    Top,      First Chapter,          , (dir)
@comment имя-ноды, следующая,    предыдущая, вверх

@menu
* Первая глава::      Первая и единственная глава в
                       этом примере.
* Указатель понятий:: В этом указателе есть два элемента.
@end menu
```

### Часть 5: Тело документа

Сегмент тела включает весь текст документа, кроме именных указателей и содержания. В этом примере показана нода и глава, содержащая нумерованный перечень.

```
@node    Первая глава, Указатель понятий, Top,     Top
@comment имя-ноды,      следующая,     предыдущая, вверх
@chapter Первая глава
@cindex Пример элемента именного указателя

Это содержимое первой главы.
@cindex Другой пример элемента именного указателя.
```

```
Это нумерованный перечень.

@enumerate
@item
Это первый пункт.

@item
Это второй пункт.
@end enumerate

Команды @code{makeinfo} и @code{texinfo-format-buffer}
преобразуют Texinfo-файлы, такие как этот, в Info-файлы;
а @TeX{} форматирует печатное руководство.
```

### Часть 6: Конец документа

Последний сегмент включает команды для создания именного указателя в отдельной ноде и ненумерованной главе и для создания содержания; в нем, также, есть команда `@bye`, которая отмечает конец документа.

```
@node    Указатель понятий,          , Первая глава, Top
@unnumbered Указатель понятий

@printindex cp

@contents
@bye
```

### Результаты

Так выглядит содержание первой главы примера:

Это содержание первой главы.

Это нумерованный перечень.

1. Это первый пункт.
2. Это второй пункт.

Команды `makeinfo` и `texinfo-format-buffer` преобразуют Texinfo-файлы, такие как этот, в Info-файлы; а TEX форматирует печатное руководство.

## 1.11 Выражение признательности

Ричард М. Столмен изобрел формат Texinfo, написал его первые процессоры и создал первую редакцию данного руководства. Роберт Дж. Чассел много пересматривал и дополнял руководство, начиная с издания 1.1. Брайн Фокс был ответственен за самомстоятельный дистрибутив Texinfo до версии 3.8 и написал отдельные программы `makeinfo` и `info`. Карл Берри делал обновления для Texinfo 3.8 и последующих выпусков, начиная с издания 2.22.

Мы благодарны всем, кто помог нам улучшить работу, особенно Франсуа Пинарду, и Девиду Д. Зуну, которые без устали записывали ошибки и неясные места и

сообщали нам о них; мы особо благодарны Мелиссе Вайссхаус за ее частые и нередко скучные обзоры похожих изданий. Неутомимые Эли Зарецкий и Андреас Шваб предоставили бесчисленное множество заплат. Зак Вайнберг сделал невозможное, реализовав синтаксис макросов в '`texinfo.tex`'. Десятки других предоставили заплаты и предложения, они с благодарностью упомянуты в файле '`ChangeLog`'. Наши ошибки — это только наши ошибки.

Немного истории: в CMU, в семидесятых Брайн Рейд разработал программу и формат, называемый Scribe, для разметки документов для печати. Для введения команд он использовал знак `@`, как и Texinfo, и старался описывать содержимое документа, а не форматирование.

Тем временем люди из MIT создали еще один, не сильно отличающийся формат, называемый Bolio. Потом он стал использовать TEX в качестве языка набора: это BoTEX.

BoTEX можно было использовать только как язык разметки для печатных документов, но не интерактивных документов. Ричард Столмен (RMS) работал как над Bolio, так и над BoTEX. Он также разработал замечательный формат интерактивной справки, называемый Info, и объединил затем BoTEX и Info в Texinfo, язык разметки для текста, предназначенный для чтения и интерактивно, и в напечатанной версии.

# 2 Использование режима Texinfo

Вы можете редактировать Texinfo-файл любым текстовым редактором, выбранным вами по вашему усмотрению. Texinfo-файл никак не отличается от любого другого ASCII-файла. Однако, при установке GNU Emacs вы получаете возможность воспользоваться специальным режимом, называемым режимом Texinfo, который снабжает вас командами Emacs и дополнительными средствами, которые могут сильно облегчить вашу работу.

Эта глава описывает только особенности режима Texinfo GNU Emacs. Здесь не рассматриваются все возможности языка форматирования документов Texinfo. Если вы читаете это руководство с самого начала, вы, вероятно, захотите быстро пролистать эту главу и вернуться к ней уже после изучения разделов, где язык форматирования документов Texinfo описывается более детально.

Режим Texinfo дает вам специальные возможности для работы с Texinfo-файлами:

- Быстрая вставка часто используемых `@`-команд.
- Автоматическое создание строк `@node`.
- Визуализация структуры Texinfo-файла.
- Автоматическое создание или изменение указателей на ноды ‘Next’, ‘Previous’ и ‘Up’.
- Автоматическое создание или модификация меню.
- Автоматическое создание главного меню.
- Форматирование части или всего файла в формат Info.
- Вывод документа на печать, печать всего файла или какой-либо его части.

Вероятно, две наиболее полезных возможности — это возможность быстрой вставки часто используемых `@`-команд и возможность создания меню и указателей на ноды.

## 2.1 Обычные команды редактирования GNU Emacs

В большинстве случаев команды режима Text работают в режиме Texinfo точно так же, как бы они работали и в самом режиме Text. В режиме Texinfo к основным возможностям редактора GNU Emacs добавляется дополнительный набор инструментов и команд редактирования. Основное различие между режимом Text и режимом Texinfo касается заполнения текста. Переменная, отвечающая за разделение абзацев, и таблица синтаксиса переопределяются в режиме Texinfo так, чтобы команды Texinfo, которые должны стоять в своих собственных строках, не были включены в другие абзацы. Поэтому команда *M-q* (`fill-paragraph`) перезаполняет абзацы, но не смешивает команды, требующие своей строки, со смежными абзацами.

Кроме того, режим Texinfo устанавливает переменную `page-delimiter` равной значению `texinfo-chapter-level-regexp`; по умолчанию, это регулярное выражение, совпадающее с командами определения глав и их аналогов, таких как приложения. С таким значением разделителя страницы вы можете переходить от одного заголовка главы к другому с помощью команд *C-x ]* (`forward-page`) и *C-x [* (`backward-page`) и сужать буфер до текста главы с помощью команды *C-x p* (`narrow-to-page`). (См.

раздел “Страницы” в *Руководство по GNU Emacs*, для более подробного описания команд, работающих со страницами.)

Вы можете называть Texinfo-файл так, как пожелаете, но по соглашению в качестве расширения следует выбрать одно из этих: ‘`.texinfo`’, ‘`.texi`’, ‘`.txi`’ или ‘`.tex`’. Для использования предпочтительно более длинное расширение, так как оно однозначно определяет тип файла. Более короткое расширение необходимо использовать в тех случаях, когда операционная система ограничивает длину имени файла. GNU Emacs автоматически входит в режим Texinfo, если вы обратитесь к файлу с расширением ‘`.texinfo`’, ‘`.txi`’ или ‘`.texi`’. Кроме того, Emacs переходит в режим Texinfo, если в теле файла в его первой строке присутствует текст ‘`-*- texinfo -*-`’. Если вы находитесь в другом режиме и желаете переключиться в режим Texinfo, выполните команду `M-x texinfo-mode`.

Естественно, как и другие возможности Emacs, режим Texinfo может быть расширен и настроен так, как вы этого пожелаете. В частности, очень легко изменить привязку ключей. Здесь описываются сделанные по умолчанию или стандартные привязки ключей.

## 2.2 Быстрая вставка часто используемых команд.

Режим Texinfo дает возможность быстро вставлять часто используемые `@`-команды в буфер редактирования. Вы можете использовать эти команды для уменьшения времени набора команд.

Чтобы вызвать команды вставки, следует дважды набрать *`C-c`* и затем нажать первый символ `@`-команды:

*`C-c C-c c`*
*`M-x texinfo-insert-@code`*
: Вставить `@code{}` и поместить точку между фигурными скобками.

*`C-c C-c d`*
*`M-x texinfo-insert-@dfn`*
: Вставить `@dfn{}` и поместить точку между фигурными скобками.

*`C-c C-c e`*
*`M-x texinfo-insert-@end`*
: Вставить `@end` и попытаться добавить правильно следующее слово, такое как ‘`example`’ или ‘`table`’. (Эта команда не может правильно обрабатывать вложенные списки, зато вставляет слово, наиболее соответствующее предшествующему списку).

*`C-c C-c i`*
*`M-x texinfo-insert-@item`*
: Вставить `@item` и поместить точку в начале следующей строки.

*`C-c C-c k`*
*`M-x texinfo-insert-@kbd`*
: Вставить `@kbd{}` и поместить точку между фигурными скобками.

*C-c C-c n*
*M-x texinfo-insert-@node*
: Вставить `@node` и добавить строку комментария, перечисляющую последовательность нод ‘Next’, ‘Previous’ и ‘Up’. Оставить точку после `@node`.

*C-c C-c o*
*M-x texinfo-insert-@noindent*
: Вставить `@noindent` и поместить точку в начале следующей строки.

*C-c C-c s*
*M-x texinfo-insert-@samp*
: Вставить `@samp{}` и поместить точку между фигурными скобками.

*C-c C-c t*
*M-x texinfo-insert-@table*
: Вставить `@table`, затем вставить ⟨SPC⟩ и поместить точку после ⟨SPC⟩.

*C-c C-c v*
*M-x texinfo-insert-@var*
: Вставить `@var{}` и поместить точку между фигурными скобками.

*C-c C-c x*
*M-x texinfo-insert-@example*
: Вставить `@example` и поместить точку в начале следующей строки.

*C-c C-c {*
*M-x texinfo-insert-braces*
: Вставить `{}` и поместить точку между фигурными скобками.

*C-c C-c }*
*C-c C-c ]*
*M-x up-list*
: Внутри пары скобок переместить точку до тех пор, пока не встретится закрывающая скобка. Набрать *C-c C-c ]* проще, чем *C-c C-c }*, что, однако, проще запоминается; поэтому для перехода между скобками существуют два ключа. (Вы также можете переходить между двумя фигурными скобками, вводя *C-f*.)

Для того, что бы вставить команду, такую как `@code{...}` вокруг *существующего* слова, надо установить точку перед этим словом и ввести последовательность *C-u 1 C-c C-c c*. Это облегчает редактирование уже существующего обычного текста. Величина параметра, следующего за *C-u*, сообщает Emacs сколько слов после точки следует заключить между фигурными скобками: ‘1’ — только одно слово, ‘2’ — два слова и так далее. Используйте отрицательное число, чтобы заключить между скобками предыдущее слово или слова. Если вы оставите параметр неопределенным, то Emacs вставит строку с `@`-командой и поместит точку между фигурными скобками. Это сработает только для тех `@`-команд, которые действуют на слово или на слова внутри одной строки, вроде `@kbd` и `@var`.

Этот набор команд вставки был создан после анализа частоты, с которой различные `@`-команды используются в *Руководстве по GNU Emacs* и в *Руководстве по GDB*. Если вы желаете добавить ваши собственные команды быстрой вставки, вы можете

привязать к ключу макрос клавиатуры, использовать сокращение или расширить код в ‘texinfo.el’.

*C-c C-c C-d* (texinfo-start-menu-description) — это команда быстрой вставки, действие которой отличается от действия других команд вставки. Она вставляет название ноды или заголовок главы в месте для описания в пункте меню. (Пункт меню состоит из трех частей: имени пункта меню, имени ноды и описания. Требуется только имя ноды, но описание сильно помогает объяснить, о чем говорится в разделе описываемой ноды. См. Раздел 7.2 [Части меню], с. 62.)

Чтобы использовать texinfo-start-menu-description, установите точку в строку пункта меню, и введите *C-c C-c C-d*. Команда ищет и копирует заголовок, который сопутствует имени ноды и вставляет этот заголовок как описание; затем устанавливает курсор в начале вставленного текста, так что вы можете отредактировать это текст. Эта функция не вставляет заголовок, если данный пункт меню уже содержит описание.

Эта команда только помогает при создании описаний; она не делает всей работы. Вы должны отредактировать вставленный текст, так как при создании заголовка обычно используют те же слова, что используются в имени ноды, но полезное описание использует другие слова.

## 2.3 Визуализация структуры разделов файла

Вы можете визуализировать структуру разделов Texinfo-файла, используя команду *C-c C-s* (texinfo-show-structure). Эта команда показывает структуру разделов Texinfo-файла, перечисляя строки, которые начинаются с @-команд, а именно, с команд @chapter, @section и подобных. Эта команда создает такую же структуру, какую представляет из себя оглавление. Эти строки отображаются в другом буфере, называемом ‘*Occur*’. В том буфере вы можете установить курсор на одной из этих строк и воспользоваться для перехода к соответствующему разделу в Texinfo-файле последовательностью *C-c C-c* (occur-mode-goto-occurrence).

*C-c C-s*
*M-x texinfo-show-structure*
: Показать строки @chapter, @section, и подобные строки Texinfo-файла.

*C-c C-c*
*M-x occur-mode-goto-occurrence*
: Перейти к строке в Texinfo-файле, соответствующей строке под курсором в буфере ‘*Occur*’.

Если вы вызываете texinfo-show-structure с префиксным аргументом, набирая *C-u C-c C-s*, это перечислит не только строки с @-командами, такими как @chapter, @section и подобными, но также и строки @node. Вы можете использовать texinfo-show-structure с префиксным аргументом для того, чтобы проверить, являются ли указатели ‘Next’, ‘Previous’ и ‘Up’ в строке @node правильными.

Часто, когда вы работаете над руководством, вам будете нужна только структура текущей главы. В этом случае вы можете выделить область буфера, структура которой вам нужна, используя команду *C-x n n* (narrow-to-region), и texinfo-show-structure будет действовать только в этой области. Чтобы снова увидеть весь буфер,

следует использовать `C-x n w` (`widen`). (См. раздел “Сужение” в *Руководство по GNU Emacs*, для получения детальной информации.)

Помимо предоставления команды визуализации структуры документа, `texinfo-show-structure`, режим Texinfo устанавливает значение переменной разделителя страницы таким образом, чтобы она соответствовала `@`-командам уровня глав. Это дает вам возможность использовать команды `C-x ]` (`forward-page`) и `C-x [` (`backward-page`) для перемещения вперед и назад по главам, а команду `C-x p` (`narrow-to-page`) — для сужения до главы. См. раздел “Страницы” в *Руководство по GNU Emacs*, для получения дополнительной информации.

## 2.4 Обновление нод и меню

В режиме Texinfo доступны команды для автоматического создания или обновления меню и указателей на ноды. Эти команды называются “update”-командами (командами обновления), потому что их основное применение — обновление Texinfo-файла после его редактирования; но вы можете использовать их, чтобы вставить указатели ‘Next’, ‘Previous’ и ‘Up’ в строку `@node`, если их там еще нет, или создать меню в файле, в котором его пока нет.

Если вы не пользуетесь командами обновления, то вам придется набирать меню и указатели на ноды вручную, а это утомительное занятие.

Вы можете использовать следующие команды обновления, чтобы

- вставить или обновить указатели на ноды ‘Next’, ‘Previous’ и ‘Up’,
- вставить или обновить меню текущего раздела и
- создать главное меню для исходного Texinfo-файла.

Вы можете использовать эти команды для обновления всех нод и меню в выделенной области или во всем Texinfo-файле.

Команды обновления работают только со стандартными Texinfo-файлами, которые имеют такую же структуру, как и книги. В таких файлах, строка с командой, объявляющей начало раздела, должна следовать сразу после каждой строки `@node`, если только это не строка `@node` для ноды ‘Top’. (*Строка с командой описания структуры глав* — это строка, начинающаяся с `@chapter`, `@section` или с другой подобной команды.)

Вы можете вставить строку с командой описания структуры глав либо вслед за строкой `@node`, либо в строке, которая следует после одиночной строки комментария `@comment`, либо после одиночной строки `@ifinfo`. Вы не можете вставить между строкой `@node` и строкой с командой описания структуры глав больше одной строки; и вставить вы можете только строку `@comment` или строку `@ifinfo`.

Команды, которые действуют на весь текст в буфере, требуют, чтобы вслед за нодой ‘Top’ следовала нода с командой `@chapter` или с эквивалентной по уровню командой. Команды обновления меню не будут создавать основное или главное меню для Texinfo-файла, который имеет ноды только уровня `@chapter`! Команды обновления меню создают меню только *внутри* нод для нод более низкого уровня. Чтобы создать меню глав, вы должны предоставить ноду ‘Top’.

Команды обновления меню удаляют пункты меню, которые относятся к другим Info-файлам, так как они не относятся к внутренним нодам текущего буфера. Это

недостаток этих команд. Вместо пунктов меню вы можете использовать перекрестные ссылки на другие Info-файлы. Ни одна из команд обновления не затрагивает перекрестные ссылки.

В режиме Texinfo существует пять наиболее часто используемых команд обновления: две из них — для обновления всех указателей нод или меню в пределах одной ноды; две — для обновления всех указателей нод или меню во всем файле; и одна команда, `texinfo-master-menu`, предназначена для создания основного или главного меню для всего файла и (что не является обязательным свойством этой команды) для обновления каждой ноды и каждого меню во всем Texinfo-файле.

Команда `texinfo-master-menu` является основной командой:

*C-c C-u m*
*M-x texinfo-master-menu*
: Создает или обновляет главное меню, которое включает в себя все другие меню (включая описания из меню, уже существующих к моменту создания основного меню, если таковые есть).

  При активизации этой команды с префиксным аргументом, с использованием последовательности *C-u*, в начале создаются или обновляются все обычные ноды и меню в текущем буфере перед созданием главного меню. (См. Раздел 3.5 [Нода Top и главное меню], с. 40, для детального описания.)

  Для функционирования `texinfo-master-menu` требуется, чтобы Texinfo-файл содержал ноду ‘Top’ и по крайней мере еще одну уровнем ниже.

  После обширного редактирования Texinfo-файла вы можете ввести следующую команду:

  ```
  C-u M-x texinfo-master-menu
  ```

  или

  ```
  C-u C-c C-u m
  ```

  Эта команда полностью обновляет все ноды и меню за один проход.

Остальные основные команды обновления выполняют работу меньшего объема и предназначены для человека, который обновляет ноду и меню в процессе создания Texinfo-файла.

Далее перечислены оставшиеся основные команды обновления:

*C-c C-u C-n*
*M-x texinfo-update-node*
: Вставляет указатели на ноды ‘Next’, ‘Previous’ и ‘Up’ для текущей ноды (то есть модифицирует ближайшую предшествующую строку `@node`). Если указатели ‘Next’, ‘Previous’ и ‘Up’ уже были в строке `@node`, то старые указатели удаляются и на их место помещаются обновленные. При запуске этой команды с аргументом (с префиксным аргументом, *C-u*, при интерактивном вызове), команда действует на все строки `@node` в выделенной области.

*C-c C-u C-m*
*M-x texinfo-make-menu*
: Создает или обновляет меню в ноде, где находится точка. При вызове этой команды с аргументом (с префиксным аргументом, *C-u*, при интерактивном вызове), команда создает или обновляет меню для нод, которые находятся внутри выделенной области.

  Всякий раз, когда `texinfo-make-menu` обновляет существующее меню, описания из этого меню включается в новое. Это достигается с помощью копирования описаний пунктов существующего меню в те пункты нового меню, которые имеют те же имена нод. Если пункты меню различаются, то описания в новое меню не копируются.

*C-c C-u C-e*
*M-x texinfo-every-node-update*
: Вставляет указатели на ноды ‘Next’, ‘Previous’ и ‘Up’ для каждой ноды в данном буфере.

*C-c C-u C-a*
*M-x texinfo-all-menus-update*
: Создает или обновляет все меню в данном буфере. С аргументом (с префиксным аргументом, *C-u*, при интерактивном вызове), перед работой над меню сначала вставляются или обновляются все указатели на ноды.

  Если существует главное меню, команда `texinfo-all-menus-update` обновляет его; но она не создает новое главное меню, если его не существует. (Для этого используйте команду `texinfo-master-menu`.)

  При работе над документом, в котором главное меню не нужно, можно сделать следующее:

  ```
  C-u C-c C-u C-a
  ```

  или

  ```
  C-u M-x texinfo-all-menus-update
  ```

  Это модифицирует все ноды и меню.

Переменная `texinfo-column-for-description` определяет позицию, по которой выровнены описания меню. По умолчанию она равна 32, хотя часто бывает полезно уменьшить ее до 24. Вы можете установить эту переменную с помощью команды *M-x edit-options*, (см. раздел “Editing Variable Values” в *Руководство по GNU Emacs*) или с помощью команды *M-x set-variable* (см. раздел “Examining and Setting Variables” в *Руководство по GNU Emacs*).

Также, команда `texinfo-indent-menu-description` может использоваться, чтобы выровнять существующие описания меню по определенной позиции. И наконец, если вы пожелаете, то вы можете использовать команду `texinfo-insert-node-lines`, чтобы вставить в файл недостающие строки `@node`. (См. Раздел 2.4.2 [Другие команды обновления], с. 22, для дополнительной информации.)

### 2.4.1 Требования для обновления

Чтобы использовать команды обновления, вы должны организовать Texinfo-файл иерархически, с главами, разделами, подразделами и так далее. Когда вы создаете

иерархию руководства, не ‘спускайтесь’ больше чем на один уровень за один раз: вслед за нодой ‘Top’ должна начинаться глава, и ни в коем случае не раздел; вслед за главой следует раздел, но не подраздел. Однако, вы можете за один раз "подняться" на любое число уровней, например от подраздела до главы.

Каждая строка `@node` за исключением строки для ноды ‘Top’ должна сопровождаться строкой с командой структурирования вроде `@chapter`, `@section` или `@unnumberedsubsec`.

Каждая комбинация (строка `@node`)/(строка команды структурирования) должна выглядеть подобно этой:

```
@node Комментарии, Минимум, Соглашения, Обзор
@comment  node-name, next,    previous,    up
@section Комментарии
```

Или подобно этой (без строки `@comment`):

```
@node Комментарии, Минимум, Соглашения, Обзор
@section Комментарии
```

В этом примере, ‘Комментарии’ — это имя и ноды, и раздела. Следующая нода называется ‘Минимум’, а предыдущая называется ‘Соглашения’. Раздел ‘Комментарии’ находится внутри ноды ‘Обзор’, которая определена как указатель‘Up’. (Вместо строки `@comment` вы можете также вписать строку `@ifinfo`.)

Если файл имеет ноду ‘Top’, то она должна назваться ‘`top`’ или ‘`Top`’ и быть первой нодой в файле.

Команды обновления меню создают меню разделов внутри главы, меню подразделов внутри раздела и так далее. Это означает, что если вы хотите получить меню глав, у вас должна быть нода ‘Top’.

Команда `makeinfo` может создать Info-файл из иерархически организованного Texinfo-файла, в котором отсутствуют указатели ‘Next’, ‘Previous’ и ‘Up’. Как следствие этого, если вы уверены, что ваш Texinfo-файл будет отформатирован с помощью `makeinfo`, команды обновления меню вам не нужны. (См. Раздел 20.1 [Создание Info-файла], с. 158, для более детальной информации о `makeinfo`.) Однако, и `makeinfo`, и команды `texinfo-format-...` требуют, чтобы вы вставили в файл меню.

### 2.4.2 Другие команды обновления

Кроме пяти основных команд обновления, режим Texinfo дает доступ к несколько менее часто используемым командам обновления:

*`M-x texinfo-insert-node-lines`*

Вставляет строки `@node` перед `@chapter` `@section`, и другими командами структурирования, если они отсутствуют.

При запуске с аргументом (с префиксным аргументом, *`C-u`*, при интерактивном вызове), команда `texinfo-insert-node-lines` не только вставляет строки `@node`, но также вставляет и названия глав или разделов в качестве имен соответствующих нод. Кроме того, она вставляет названия в качестве имен нод в уже существующие строки `@node`, не имеющие имен. Так как имена нод должны быть более краткими, чем названия разделов или глав, вы должны вручную отредактировать вставленные имена.

Например, для того, чтобы выделить весь буфер и везде вставить строки `@node` и названия, следует сделать это:

```
C-x h C-u M-x texinfo-insert-node-lines
```

Эта команда вставляет в качестве имен нод в строках `@node` названия разделов; команда `texinfo-start-menu-description` (см. Раздел 2.2 [Быстрая вставка], с. 16) вставляет названия как описания пунктов меню — это разные действия. Однако в обоих случаях вы должны отредактировать вставленный текст.

*M-x texinfo-multiple-files-update*
: Обновляет ноды и меню в документе, сформированном из отдельных файлов. При запуске с *C-u* создает и вставляет главное меню во внешний файл. При запуске с числовым аргументом, как *C-u 2*, перед созданием и вставкой главного меню во внешнем файле сначала модифицирует все меню и все указатели ‘Next’, ‘Previous’ и ‘Up’ во всех файлах. Команда `texinfo-multiple-files-update` описана в приложении о включаемых файлах. См. Раздел E.2 [texinfo-multiple-files-update], с. 202.

*M-x texinfo-indent-menu-description*
: Выравнивает каждое описание в меню после точки по определенному столбцу. Вы можете использовать эту команду, чтобы создать для описаний больше места. При запуске с аргументом (с префиксным аргументом, *C-u*, при интерактивном вызове), команда `texinfo-indent-menu-description` выравнивает каждое описание в каждом меню в выделенной области. Однако, эта команда не выравнивает вторые и последующие строки многострочного описания.

*M-x texinfo-sequential-node-update*
: Вставляет имена нод, находящихся непосредственно перед и после текущей ноды как ‘Next’ или ‘Previous’, независимо от иерархического положения этих нод. Это означает, что следующая нода подраздела может быть следующей главой. Последовательная структура нод полезна для романов и других документов, которые следует читать последовательно. (Однако, команда Info *g ** позволяет вам просматривать файл последовательно, а значит, такая упорядоченность нод не является строго необходимой). При запуске с аргументом, команда `texinfo-sequential-node-update` последовательно обновляет все ноды в области.

## 2.5 Форматирование для Info

Режим Texinfo обеспечивает отдельные команды для форматирования части или всего Texinfo-файла для Info. Часто, когда вы пишете документ, вы хотите отформатировать только часть файла — то есть некую область.

Вы можете использовать команду `texinfo-format-region` или `makeinfo-region`, чтобы отформатировать область:

*C-c C-e C-r*
*M-x texinfo-format-region*
*C-c C-m C-r*
*M-x makeinfo-region*
: Форматирует текущую область для Info.

Вы можете использовать команду `texinfo-format-buffer` или `makeinfo-buffer` для того, чтобы отформатировать весь буфер:

*C-c C-e C-b*
*M-x texinfo-format-buffer*
*C-c C-m C-b*
*M-x makeinfo-buffer*
: Форматирует текущий буфер для Info.

Например, после создания Texinfo-файла вы можете набрать следующее:

```
C-u C-c C-u m
```

или

```
C-u M-x texinfo-master-menu
```

Это модифицирует все ноды и меню. Затем наберите следующее, чтобы создать Info-файл:

```
C-c C-m C-b
```

или

```
M-x makeinfo-buffer
```

Для того, чтобы могли работать TEX или команды форматирования для Info, в заголовке файла *должна* быть строка `@setfilename`.

См. Раздел 20.1 [Создание Info-файла], с. 158, для более подробного описания форматирования для Info.

## 2.6 Форматирование и печать

Набор и печать Texinfo-файла — процесс многоступенчатый, в котором вы сначала создаете файл для печати (называемый DVI-файлом) и затем печатаете его. Кроме того, вы можете также создавать именные указатели. Чтобы сделать это, вы должны выполнить команду `texindex` после первичного выполнения команды `tex`; а затем выполнить команду `tex` снова. Или следует выполнить команду `texi2dvi`, которая сама автоматически создает именные указатели. (см. Раздел 19.3 [Форматирование с texi2dvi], с. 148).

Часто, когда вы пишете документ, вы хотите вывести на печать только часть файла, чтобы увидеть, как она будет выглядеть. Для этой цели вы можете использовать команду `texinfo-tex-region` или похожие команды. Используйте команду `texinfo-tex-buffer`, чтобы отформатировать весь буфер.

*C-c C-t C-b*
*M-x texinfo-tex-buffer*
: Выполняет команду `texi2dvi` для буфера. Помимо запуска TEX для буфера, эта команда автоматически создает или обновляет именные указатели.

*C-c C-t C-r*
*M-x texinfo-tex-region*
: Выполняет команду TEX для области.

*C-c C-t C-i*
*M-x texinfo-texindex*
: Запускает `texindex`, чтобы отсортировать именные указатели Texinfo-файла, отформатированного с помощью команды `texinfo-tex-region`. Команда `texinfo-tex-region` не запускает `texindex` автоматически; она только выполняет команду `tex`. Вы должны запустить команду `texinfo-tex-region` во второй раз после сортировки необработанного именных указателей командой `texindex`. (Обычно, когда форматируют область, именные указатели не формируют; это делается только для всего буфера. Сейчас, когда существует команда `texi2dvi`, обсуждаемая команда практически не нужна.)

*C-c C-t C-p*
*M-x texinfo-tex-print*
: Печатает файл (или часть файла), предварительно отформатированный с помощью команды `texinfo-tex-buffer` или `texinfo-tex-region`.

Для того, чтобы команда `texinfo-tex-region` или `texinfo-tex-buffer` сработала, файл *должен* начинаться со строки '`\input texinfo`' и должен включать строку `@settitle`. Файл должен заканчиваться командой `@bye`, набранной в отдельной строке. (Когда вы используете команду `texinfo-tex-region`, вы должны окружить `@settitle` строками start-of-header и end-of-header.)

См. Глава 19 [Печать], с. 146, для подробного описания других команд для TEX, вроде `tex-show-print-queue`.

## 2.7 Резюме по режиму Texinfo

Каждый набор команд режима Texinfo имеет привязки по умолчанию, начинающиеся с одних ключей. Все команды, созданные специально для режима Texinfo, начинаются с *C-c*. Ключи отчасти мнемонические.

### Команды вставки

Команды вставки вызываются с помощью набора двух *C-c* и затем первого символа вставляемой `@`-команды. (Вероятно правильнее было бы для мнемоничности использовать *C-c C-i*, от 'custom insert', но *C-c C-c* набирается гораздо быстрее.)

| | |
|---|---|
| `C-c C-c c` | Вставить '`@code`'. |
| `C-c C-c d` | Вставить '`@dfn`'. |
| `C-c C-c e` | Вставить '`@end`'. |
| `C-c C-c i` | Вставить '`@item`'. |
| `C-c C-c n` | Вставить '`@node`'. |
| `C-c C-c s` | Вставить '`@samp`'. |
| `C-c C-c v` | Вставить '`@var`'. |
| `C-c C-c {` | Вставить фигурные скобки. |
| `C-c C-c ]` | |

`C-c C-c }` Перейти за пределы огражденного скобками участка.
`C-c C-c C-d` Вставить название раздела для ноды в месте для описания в пункте меню.

## Визуализация структуры

Команда `texinfo-show-structure` часто используется внутри суженной области.

`C-c C-s` Перечислить все заголовки.

## Команда обновления главного меню

Команда `texinfo-master-menu` создает главное меню; может использоваться для того, чтобы модифицировать каждую ноду и каждое меню в файле.

`C-c C-u m M-x texinfo-master-menu` Создать или обновить главное меню.
`C-u C-c C-u m` С *C-u* в качестве префиксного аргумента, сначала создать или обновить все ноды и обычные меню, а затем создать главное меню.

## Обновление указателей

Команды обновления указателей вызываются при наборе *C-c C-u* и затем или *C-n* для `texinfo-update-node`, или *C-e* для `texinfo-every-node-update`.

`C-c C-u C-n` Обновить ноду.
`C-c C-u C-e` Обновить все ноды в буфере.

## Обновление меню

Команды обновления меню вызываются при наборе *C-c C-u*, и затем или *C-m* для `texinfo-make-menu`, или *C-a* для `texinfo-all-menus-update`. Чтобы обновить и ноды, и меню в одно и то же время, перед набором командной последовательности *C-c C-u C-a* наберите *C-u*.

`C-c C-u C-m` Создать или обновить меню.
`C-c C-u C-a` Создать или обновить все меню в буфере.
`C-u C-c C-u C-a` С префиксным аргументом *C-u*, сначала создать или обновить все ноды, а затем создать или обновить все меню.

## Форматирование для Info

Команды форматирования для Info, которые написаны на языке Emacs Lisp, вызываются при наборе *C-c C-e* и затем или *C-r* для выделенной области, или *C-b* для всего буфера.

Команды форматирования для Info, которые написаны на Си и основаны на программе `makeinfo`, вызываются при наборе *C-c C-m* и затем или *C-r* для выделенной области, или *C-b* для всего буфера.

Использование команды `texinfo-format...`:

`C-c C-e C-r` Форматировать область.
`C-c C-e C-b` Форматировать буфер.

Использование `makeinfo`:

`C-c C-m C-r` Форматировать область.
`C-c C-m C-b` Форматировать буфер.
`C-c C-m C-l` Обновить выходной буфер `makeinfo`.
`C-c C-m C-k` Прекратить выполнение `makeinfo`.

## Вывод и печать

Команды набора и печати через TEX вызываются при наборе последовательности *`C-c C-t`* и затем другой управляющей последовательности: *`C-r`* для `texinfo-tex-region`, *`C-b`* для `texinfo-tex-buffer` и так далее.

`C-c C-t C-r` Выполнить TEX для области.
`C-c C-t C-b` Выполнить `texi2dvi` для буфера.
`C-c C-t C-i` Выполнить `texindex`.
`C-c C-t C-p` Напечатать DVI-файл.
`C-c C-t C-q` Показать очередь заданий на принтер.
`C-c C-t C-d` Удалить задание из очереди заданий на принтер.
`C-c C-t C-k` Прекратить выполнение команды форматирования в TEX.
`C-c C-t C-x` Выйти из приостановленного задания форматирования в TEX.
`C-c C-t C-l` Обновить выходной буфер.

## Другие команды обновления

Для вызова остальных команд обновления нет стандартных ключей, потому что они используются редко.

`M-x texinfo-insert-node-lines`
Вставить пропущенные строки `@node` в области. С префиксным аргументом *`C-u`*, использует названия разделов, как имена нод.

`M-x texinfo-multiple-files-update`
Обновить документ, состоящий из нескольких файлов. С префиксным аргументом *`C-u 2`*, создать или обновить все ноды или меню во всех файлах документа.

`M-x texinfo-indent-menu-description`
Выровнять описания.

`M-x texinfo-sequential-node-update`
Вставить указатели на ноды в строгой последовательности.

# 3 Начало Texinfo-файла

В начале Texinfo-файла должна предоставляться определенная информация, вроде имени файла и названия документа.

Обычно начальная информация Texinfo-файла состоит из четырех частей:

1. Заголовок, отделенный специальными строками комментария, который включает в себя команды для определения названия Texinfo-файла и сообщение для TEX, в котором указывается какой файл определений нужно использовать при обработке Texinfo-файла.
2. Краткое сообщение о том, какую информацию можно найти в этом файле, авторские права и разрешения. Это заключено между командами `@ifinfo` и `@end ifinfo`, чтобы программа форматирования поместила это сообщение в Info-файл.
3. Титульный лист и страница с информацией об авторских правах, с сообщением об авторских правах и разрешениями на распространение. Эта часть заключается между командами `@titlepage` и `@end titlepage`. Титульный лист и страница с информацией об авторских правах появляется только при печати руководства.
4. Первая нода, в которой содержится меню для всего Info-файла. Содержимое этой части появляется только в Info-файле.

Также, по желанию вы можете включить условия распространения и отказ от предоставления гарантий для программы. Информация об условиях распространения и гарантиях обычно следует за введением или включается в первую главу руководства.

Так как текст сообщения об авторских правах и условий распространения для документа Texinfo (в отличие от условий распространения для программы) находится в частях, которые появляется либо только в Info-файле, либо только в печатном руководстве, эту информацию следует дублировать.

## 3.1 Образец начала Texinfo-файла

В следующем примере показано что следует вставить в начало Texinfo-файла.

```
\input texinfo   @c -*-texinfo-*-
@c %**start of header
@setfilename имя-texinfo-файла
@settitle название-руководства
@setchapternewpage odd
@c %**end of header

@ifinfo
Этот файл описывает ...

(C) обладатель авторских прав, год

Предоставляется разрешение ...
@end ifinfo

@c Этот пример иллюстрирует только один из
@c двух методов формирования титульного листа.
```

```
@titlepage
@title название-печатного-руководства
@subtitle подзаголовок-если-нужен
@subtitle второй-подзаголовок
@author автор

@c Со следующих двух команд начинается страница
@c с информацией об авторских правах.
@page
@vskip 0pt plus 1filll @copyright{} год
обладатель авторских прав

Опубликовано ...

Предоставляется разрешение ...
@end titlepage

@node Top, Обзор, , (dir)

@ifinfo
В этом документе описывается ...

Этот документ соответствует версии ...
программы ...
@end ifinfo

@menu
* Авторские права::           Ваши права и возможности.
* Первая глава::              Начало ...
* Вторая глава::              ...
...
...
@end menu

@node   Первая глава, Вторая глава, top,      top
@comment node-name,     next,           previous, up
@chapter Первая глава
@cindex Вхождения именных указателей для первой главы
```

## 3.2 Заголовок Texinfo-файла

Texinfo-файл начинается по крайней мере с трех строк, которые предоставляют необходимую информацию для Info и TEX. Это строка `\input texinfo`, строка `@settitle` и строка `@setfilename`. Если вы хотите, чтобы TEX отформатировал только часть Texinfo-файла, вы должны вставить строки `@settitle` и `@setfilename` между строк start-of-header и end-of-header.

Таким образом, начало Texinfo-файла выглядит так:

```
\input texinfo   @c -*- texinfo -*-
@setfilename пример.info
@settitle Пример оформления документа
```

или так:

```
\input texinfo   @c -*- texinfo -*-
@c %**start of header
@setfilename Пример.info
@settitle Пример оформления документа
@c %**end of header
```

### 3.2.1 Первая строка Texinfo-файла

Каждый Texinfo-файл, если его предполагается обрабатывать с помощью низкоуровневых процедур форматирования TEX, должен начинаться со строки подобной этой:

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

Эта строка служит для двух целей:

1. Когда файл обрабатывается TEX, команда '`\input texinfo`', сообщает TEX о том, что следует загружать макросы, необходимые для обработки Texinfo-файла. Эти макросы находятся в файле, называемом '`texinfo.tex`', который обычно располагается в каталоге '`/usr/lib/tex/macros`'. TEX использует обратную косую черту, '\', как отметку начала команды, так же, как Texinfo использует '`@`'. В файле '`texinfo.tex`' сигнальный символ '\' переопределяется на '`@`'; до этого переключения TEX требует '\', поэтому эта строка должна находится в самом начале файла.
2. Когда файл редактируется в GNU Emacs, строка '`-*- texinfo -*-`' сообщает Emacs, что следует использовать режим Texinfo.

### 3.2.2 Начало заголовка

Вставьте строку start-of-header второй строкой в Texinfo-файле. За строкой start-of-header следуют строки, начинающиеся с команд `@setfilename` и `@settitle`, а иногда с команд `@smallbook` или `@footnotestyle`. Заголовок замыкается строкой end-of-header (см. Раздел 3.2.8 [Конец заголовка], с. 34).

С помощью этих строк вы можете форматировать часть Texinfo-файла для Info или для выдачи на печать.

Строка start-of-header выглядит следующим образом:

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

Странная последовательность знаков '`%**`' должна гарантировать, что никакой другой комментарий не будет случайно перепутан с первой строкой заголовка.

### 3.2.3 `@setfilename`

Чтобы Texinfo-файл можно было отформатировать с помощью `makeinfo` или TEX, он должен содержать следующую строку:

`@setfilename` *имя-info-файла*

Команду `@setfilename` следует вставлять в начале строки, затем нужно вставить имя Info-файла. Не следует писать на этой строке что-либо еще; все, что следует после этой команды, рассматривается как имя Info-файла, даже то, что в ином другом случае воспринималось бы как комментарий.

Строка `@setfilename` определяет имя создаваемого при форматировании выходного файла. Это имя должно отличаться от имени Texinfo-файла. Существуют два соглашения для выбора имени файла, вы можете или удалить из имени файла расширение (такое как '`.texi`'), или заменять расширение на '`.info`'. При создании HTML-файлов `makeinfo` заменяет любое расширение на '`html`' или добавляет '`.html`', если у заданного файла нет расширения.

Некоторые операционные системы не поддерживают длинные имена файлов. У вас могут возникнуть проблемы даже тогда, когда имя файла достаточно коротко. Это происходит, потому что программа форматирования Info-файлов разбивает длинный файл на более короткие и дает им имена путем добавления в конец префиксов '`-1`', '`-2`', ..., '`-10`', '`-11`' и так далее. (См. Раздел 20.1.8 [Теги и разбиение файлов], с. 165.) Таким образом получаются имена файлов вроде '`texinfo.info-10`', которые для некоторых систем являются слишком длинными; так что подобные файлы лучше назвать '`texinfo`', а не '`texinfo.info`'. Когда программа `makeinfo` выполняется на операционных системах вроде MS-DOS, которые налагают серьезные ограничения на длину имен файла, то она иногда будет укорачивать первоначальное имя файла, чтобы обеспечить для суффиксов достаточно места, производя таким образом файлы с названиями вроде '`texin-10`', '`gcc.i12`' и так далее.

Команды форматирования для Info игнорируются, если они находятся перед строкой `@setfilename`, именно поэтому самая первая строка в файле (строка `\input`) не появляется при выводе.

Строка `@setfilename` не производит никакого вывода при форматировании руководства с помощью TEX, но несмотря на это она необходима: она открывает вспомогательные файлы для именных указателей, перекрестных ссылок и другие вспомогательные файлы, используемые Texinfo, а также считывает файл '`texinfo.cnf`', если он присутствует в системе (см. Раздел 19.9 [Подготовка к применению TEX], с. 152).

### 3.2.4 `@settitle`

Чтобы из Texinfo-файла можно было создать печатное руководство, он должен содержать следующую строку:

`@settitle` *Заголовок*

Команда `@settitle` вставляется в начале строки, а за ней на этой же строке следует заголовок. Она сообщает TEX заголовок, который следует печатать вверху и внизу страницы. Больше в этой строке ничего писать не следует; все что находится в этой строке, даже комментарий, становится заголовком.

По традиции, когда TEX форматирует Texinfo-файл для двусторонней печати, заголовок печатается вверху на левых (четных) страницах, а название текущей главы — вверху на правых (нечетных) страницах. (TEX получает название каждой главы из команды `@chapter`). Внизу страницы ничего не печатается.

Даже если вы печатаете в одностраничном стиле, TEX ищет строку с командой `@settitle`, если вы включаете название документа в заголовок.

Команда `@settitle` должна предшествовать всему, что производит действительный вывод в TEX.

Хотя заголовок в команде `@settitle` обычно тот же самый, что и заголовок на титульном листе, это никак не влияет на то, что печатается на титульном листе. Таким

образом, эти два заголовка не обязаны совпадать; заголовок в команде `@settitle` может быть сокращенной или расширенной версией названия, печатаемого на титульном листе. (См. Раздел 3.4.1 [`@titlepage`], с. 35.)

TEX печатает заголовки страниц только для того текста, который следует в Texinfo-файле после команды `@end titlepage` или находится после команды `@headings`, которая включает печать заголовков. См. Раздел 3.4.6 [Команда `@headings`], с. 39, для более детальной информации.

Вы также можете, если желаете, создать ваши собственные заголовки страниц. Для более детальной информации смотрите Приложение F [Заголовки страниц], с. 206.

### 3.2.5 `@setchapternewpage`

В официально изданной книге текст обычно печатается на обеих сторонах бумаги, главы начинаются на правых страницах, и правые страницы имеют нечетные номера. Но в коротких докладах текст часто печатается только на одной стороне бумаги. Также в коротких докладах главы иногда не начинаются на новых страницах, а печатаются на той же самых странице, где был конец предшествующей главы, и отделяются от него небольшим вертикальным отступом.

Вы можете использовать команду `@setchapternewpage` с различными аргументами, чтобы определить, как TEX должен начинать главы и должен ли он форматировать текст для печати на одной или на обеих сторонах бумаги (односторонняя или двусторонняя печать).

Пишите команду `@setchapternewpage` в начале строки, далее добавьте соответствующий аргумент.

Например, вы хотите, чтобы все главы начинались с новой нечетной страницы:

```
@setchapternewpage odd
```

С помощью команды `@setchapternewpage` вы можете выбрать одну из трех альтернатив:

`@setchapternewpage off`
: Заставляет TEX начинать новую главу на той же самой странице, где кончается предыдущая глава после некоторого вертикального пропуска. Также, заставляет TEX форматировать заголовки страниц для односторонней печати. (Вы можете переопределить формат заголовков с помощью команды `@headings double`. Для дополнительной информации смотрите Раздел 3.4.6 [Команда `@headings`], с. 39.)

`@setchapternewpage on`
: Заставляет TEX начинать новые главы с новых страниц и форматировать заголовки страниц для односторонней печати. Эта форма наиболее часто используется для коротких докладов или личных распечаток.

  Эта альтернатива осуществляется по умолчанию.

`@setchapternewpage odd`
: Заставляет TEX начинать новые главы с новых нечетных (правых) страниц и форматировать текст для двусторонней печати. Это форма наиболее часто используется для книг и руководств.

В Texinfo нет команды `@setchapternewpage even`.

Вы можете переопределить или изменить действие команды `@setchapternewpage` на заголовки с помощью команды `@headings`. См. Раздел 3.4.6 [Команда @headings], с. 39.)

В начале руководства или книги страницы не нумеруются; например, в книгах не нумеруются титульный лист и страница с информацией об авторских правах. В соответствии с соглашениями страницы содержания нумеруются римскими цифрами и выпадают из нумерации остальной части документа.

Так как Info-файл не имеет страниц, то команда `@setchapternewpage` не влияет на него.

Мы рекомендуем вообще не включать команду `@setchapternewpage` в исходный текст вашего руководства, так как желаемый вывод не определяется внутренне документом. Вместо этого, если вам не нравится вариант по умолчанию (нет пустых строк, на всех страницах одинаковые заголовки), используйте ключ '`-texinfo`' для команды `texi2dvi`, чтобы указать, какой вывод вы хотите.

### 3.2.6 Отступ в начале абзаца

Texinfo-процессоры могут вставить несколько пробелов перед началом абзаца в первой его строке, выделяя таким образом этот абзац. Вы можете использовать команду `@paragraphindent`, чтобы определить величину отступа. Пишите команду `@paragraphindent` в начале строки, сопровождая ее параметром '`asis`' или числом, как здесь:

`@paragraphindent` *отступ*

Отступ делается в соответствии со значением числа *отступ*:

`asis` Не изменять существующий отступ (не реализовано в TEX).

0 Не делать отступ.

*n* Создавать отступ размером в *n* пробелов при выводе в Info и в *n* единиц em в TEX.

Значение переменной *отступ* по умолчанию равно '`asis`'. `@paragraphindent` игнорируется при выводе в формате HTML.

Вставляйте команду `@paragraphindent` перед или вскоре после строки end-of-header в начале Texinfo-файла. (Если вы вставите команду между строк start-of-header и end-of-header, то команды форматирования области будут делать для абзацев указанный отступ.)

Особенность команд `texinfo-format-buffer` и `texinfo-format-region` в том, что они не делают отступ (и не заполняют текст) в тех абзацах, которые содержат команды `@w` или `@*`. См. Приложение H [Перезаполнение абзацев], с. 219, для получения дальнейшей информации.

### 3.2.7 `@exampleindent`: отступы в блоках

Texinfo-процессоры делают отступ в каждой строке блоков `@example` и подобных ему. Для задания этого отступа вы можете использовать команду `@exampleindent`.

Пишите команду `@exampleindent` на отдельной строке, а после нее указывайте число или ‘`asis`’:

`@exampleindent` *отступ*

Отступ делается в соответствии со значением параметра *отступ*:

`asis` Не изменять существующие отступы (не реализовано в TEX).

0 Опустить отступы.

*n* Создавать отступ из *n* пробелов в Info-выводе или из *n* единиц em в TEX.

Значение *отступа* по умолчанию равно пяти. `@exampleindent` игнорируется при выводе в HTML.

Пишите команду `@exampleindent` перед или немного после строки, end-of-header, завершающей заголовок Texinfo-файла. (Если вы напишете эту команду между строк start-of-header и end-of-header, команды форматирования области будут делать в примерах указанный отступ.)

### 3.2.8 Окончание заголовка

Заканчивайте заголовок строкой end-of-header. Эта строка выглядит следующим образом:

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

Если в заголовке файла, между строк start-of-header и end-of-header, вы вставляете команду `@setchapternewpage`, TEX будет форматировать область в соответствии с этой командой. Точно так же, если вы включаете между строк start-of-header и end-of-header команду `@smallbook`, TEX будет форматировать область в формате “маленькая книга”.

См. Раздел 3.2.2 [Начало заголовка], с. 30.

## 3.3 Обзор и разрешения на копирование для Info

Титульный лист и страница с информацией об авторских правах появляются только в печатной копии руководства; поэтому та же самая информация должна быть вставлена в секцию, которая появляется только в Info-файле. Эта секция обычно содержит краткое описание содержания Info-файла, уведомление об авторских правах и правах на распространение.

Уведомление об авторских правах должно выглядеть так:

© *обладатель-авторских-прав, год*

и находиться на отдельной строке.

Стандартный текст уведомления содержится в приложении к этому руководству; смотрите Раздел D.1 [‘`ifinfo`’ Разрешения на копирование], с. 199, для получения полной версии текста.

Текст условий распространения появляется в Info-файле *перед* первой нодой. Это означает, что читатель *не* видит этот текст при чтении файла с использованием Info, если только он не использует продвинутую команду Info *g**.

## 3.4 Титульный лист и страница с информацией об авторских правах

Название руководства и имя автора обычно печатаются на титульном листе.

Иногда на титульном листе также печатается информация об авторских правах; обычно же информация об авторских правах печатается на обратной стороне титульного листа. Титульный лист и страница с информацией об авторских правах появляются в печатном руководстве, но не в Info-файле. Из-за этого возможно использование некоторых малопонятных команд TEX, которые не могут использоваться в Info-файле. Кроме того, эта часть начала Texinfo-файла содержит текст условий распространения, который появится в печатном руководстве.

Вы можете захотеть включить информацию вида титульного листа в вывод в простом текстовом формате. Просто поместите любой такой начальный материал между `@ifinfo` и `@end ifinfo`; `makeinfo` включит его в простой текстовый вывод. Этого не будет видно в программах чтения Info.

См. Раздел D.2 [Разрешения на титульном листе], с. 200, стандартный текст условий распространения.

### 3.4.1 `@titlepage`

Начинайте материал для титульного листа и последующей страницы с информацией об авторских правах с команды `@titlepage`, занимающей всю первую строку, и заканчивайте строкой с командой `@end titlepage`.

Команда `@end titlepage`, начинает новую страницу и включает нумерацию страниц. (См. Приложение F [Заголовки страниц], с. 206, для дополнительной информации). Весь материал, который вы хотите разместить на ненумерованных страницах, должен быть помещен между командами `@titlepage` и `@end titlepage`. Вы можете создать в этом месте содержание с помощью команды `@setcontentsaftertitlepage` (см. Раздел 4.2 [Содержание], с. 44).

Используйте команду `@page`, чтобы вставить в пределах области, очерченной командами `@titlepage` и `@end titlepage`, разрыв страницы и создать таким образом больше одной ненумерованной страницы. Так как создается страница с информацией об авторских правах. (Вероятно команду `@titlepage` лучше было бы назвать `@titleandadditionalpages`, но это слишком длинно!)

Когда вы пишете руководство по компьютерной программе, вы должны указать на титульном листе версию программы, к которой применимо данное руководство. Если руководство изменяется чаще, чем программа, или независимо от нее, вы должны также включить номер редакции[1]. Это помогает читателям понимать, которое руководство для какой версии программы. (Первая нода также должна содержать эту информацию; смотрите Раздел 5.3 [`@top`], с. 47.)

Texinfo предоставляет два основных метода для создания титульного листа. Один метод использует команды `@titlefont`, `@sp` и `@center`, чтобы создать титульный лист, в котором слова на странице центрируется.

[1] Мы нашли, что полезно обращаться к версиям руководств как “редакция”, а к версиям программ как “версия”; иначе, говоря одними словами и о программах, и о документации, мы вводим друг в заблуждение.

Второй метод использует команды `@title`, `@subtitle` и `@author`, чтобы создать титульный лист с черными линейками под названием и именем автора и текстом подзаголовка, прижатыми к правому краю страницы. При использовании этого метода, вы не определяете никакого форматирования для титульного листа. Вы задаете желаемый текст, а Texinfo делает форматирование.

Вы можете использовать один из методов или комбинировать их; смотрите последующие разделы.

Для чрезвычайно простых случаев в Texinfo есть команда `@shorttitlepage`, которая принимает единственный аргумент, название. Аргумент набирается на отдельной странице, а затем следует чистая страница.

### 3.4.2 `@titlefont`, `@center` и `@sp`

Чтобы создать титульный лист для печатного документа, вы можете использовать команды `@titlefont`, `@sp` и `@center`. (Это первый из двух методов для создания титульного листа в Texinfo.)

Используйте команду `@titlefont`, чтобы выбрать больший шрифт, подходящий для названия. Вы можете использовать `@titlefont` больше одного раза, если у вас особенно длинное название.

Например:

```
@titlefont{Texinfo}
```

Используйте команду `@center` в начале строки, чтобы отцентрировать текст в этой строке. Например:

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

центрирует название, которым в этом примере является “Texinfo”, напечатанное шрифтом для названий.

Используйте команду `@sp`, чтобы вставить вертикальный пробел. Например:

```
@sp 2
```

Это вставляет две незаполненные строки на напечатанной странице. (См. Раздел 14.4 [@sp], с. 119, более подробную информацию о команде `@sp`.)

Шаблон для этого способа выглядит так:

```
@titlepage
@sp 10
@center @titlefont{имя-руководства-при-печати}
@sp 2
@center подзаголовок-если-есть
@sp 2
@center автор
...
@end titlepage
```

Промежутки в примере такие, что текст вмещается на странице формата 8.5 на 11 дюймов.

### 3.4.3 `@title`, `@subtitle` и `@author`

Вы можете использовать команды `@title`, `@subtitle` и `@author` для создания титульного листа, в котором вертикальный и горизонтальный интервал делаются для вас автоматически. Это отличается от метода, описанного выше, в котором команда `@sp` является необходимой для настройки вертикальных пропусков.

Пишите команды `@title`, `@subtitle` и `@author` в начале строки, далее за ними в этих же строках должны следовать заголовок, подзаголовок и автор, соответственно.

Команда `@title` создает строку, в которой название прижимается к левому краю страницы и печатается более крупным, чем обычно, шрифтом. Название подчеркивается черной линейкой. Допустима только одинарная строка; нельзя использовать команду `@*`, чтобы разбить название на две строки. Чтобы иметь возможность работать с длинными названиями, вам может показаться полезным использовать обе команды `@title` и `@titlefont`; смотрите заключительный пример в этом разделе.

Команда `@subtitle` набирает подзаголовок шрифтом обычного размера и прижимает его к правому краю страницы.

Команда `@author` печатает шрифтом среднего размера имя автора или авторов в левой части страницы недалеко от нижнего края титульного листа. Имена подчеркиваются черной линейкой, которая тоньше, чем та, что подчеркивает заголовок. (Черная линейка появляется только в том случае, если вслед за строкой команды `@author` следует строка команды `@page`.)

Есть два способа применения команды `@author`: вы можете ввести имя или имена на остающейся части строки, которая начинается с `@author`:

```
@author Джейн Смит и Джон До
```

или вы можете ввести имена одно над другим, используя две (или больше) команды `@author`:

```
@author Джейн Смит
@author Джон До
```

(Только нижнее имя подчеркивается черной линейкой.)

Шаблон для этого метода выглядит так:

```
@titlepage
@title имя-руководства-при-печати
@subtitle подзаголовок-если-есть
@subtitle второй-подзаголовок
@author автор
@page
...
@end titlepage
```

Вы можете также комбинировать метод `@titlefont`, описанный в предыдущем разделе и метод `@title`, описанный в этом. Это может быть полезно, если у вас есть очень длинное название. Вот пример из реальной жизни:

```
@titlepage
@titlefont{GNU Software}
@sp 1
@title for MS-Windows and MS-DOS
@subtitle Edition @value{edition} for Release @value{cd-edition}
@author by Daniel Hagerty, Melissa Weisshaus
@author and Eli Zaretskii
```

(Такое использование `@value` объясняется в Раздел 16.4.3 [Пример использования @value], с. 138.)

### 3.4.4 Страница с информацией об авторских правах и разрешения

В соответствии с международным соглашением, текст уведомления об авторских правах для книги должен находиться или на титульном листе, или на обратной стороне титульного листа. Текст этого уведомления должен включать год и название организации или имя человека, кому принадлежат авторские права.

Когда текст авторского права печатается на обратной стороне титульного листа, эта страница обычно не нумеруется. Значит, текст уведомления об авторских правах должен писаться в Texinfo между команд `@titlepage` и `@end titlepage`.

Используйте команду `@page`, чтобы разорвать страницу. Для того, чтобы сдвинуть текст уведомления об авторских правах и остальной текст на странице с информацией об авторских правах, вы можете написать после команды `@page` несколько таинственную строку, выглядящую так:

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

Это команда TEX, которая не поддерживается командами форматирования для Info. Команда `@vskip` вставляет вертикальный пропуск. Команда '`0pt plus 1filll`' заставляет TEX создать такой вертикальный пропуск, чтобы сместить последующий текст к нижнему краю страницы. Обратите внимание на использование трех '`l`' в слове '`filll`', это правильно в TEX.

В печатном руководстве команда `@copyright{}` создает знак охраны авторских прав — '`c`' внутри круга. (В Info этот знак воспроизводится как '`(C)`'.) Уведомление об авторских правах имеет официально утвержденную форму записи:

© *обладатель-авторских-прав, год*

Принято помещать информацию о том, как получить руководство, сразу после уведомления об авторских правах перед разрешением на распространение руководства.

Разрешение на распространение должно быть записано здесь, а так же в сегменте обзора между `@ifinfo` и `@end ifinfo` сразу после заголовка, так как этот текст появляется только в печатном руководстве, а текст из блока '`ifinfo`' появляется только в Info-файле.

См. Приложение D [Пример разрешений], с. 199, стандартный текст разрешений.

### 3.4.5 Создание заголовков

Команда `@end titlepage`, занимающая целую строку, не только отмечает конец титульного листа и страницы с информацией об авторских правах, но и заставляет TEX создавать заголовки и номера страниц.

Как уже было сказано, Texinfo имеет два стандартных формата заголовков страниц, один для документов, которые печатаются на одной стороне каждого листа бумаги (односторонняя печать), и другой для документов, которые печатаются с обеих сторон каждого листа (двусторонняя печать). (См. Раздел 3.2.5 [@setchapternewpage], с. 32.) Вы можете задать эти форматы различными способами:

- Обычный путь состоит в том, чтобы вставить команду `@setchapternewpage` до команд титульного листа и затем написать команду `@end titlepage`, с которой уже начнется создание заголовков страниц в желаемом вами стиле. (См. Раздел 3.2.5 [@setchapternewpage], с. 32.)
- Или вы можете использовать команду `@headings`, чтобы предотвратить создание заголовков страниц или начинать их создание для или односторонней или двусторонней печати. (Пишите команду `@headings` сразу после команды `@end titlepage`. См. Раздел 3.4.6 [Команда @headings], с. 39, для дополнительной информации.)
- Или вы можете определить ваш собственный формат верхних и нижних заголовков страниц. См. Приложение F [Заголовки страниц], с. 206, для более подробной информации о верхних и нижних заголовках страниц.

Большинство документов форматируются в стандартном одностороннем или двустороннем формате. Для печати в двустороннем формате используется команда `@setchapternewpage odd`, а односторонняя печать производится, если команда `@setchapternewpage` не задана.

### 3.4.6 Команда `@headings`

Команда `@headings` используется редко. Она задает вид верхних и нижних заголовков страниц. Обычно этим управляет команда `@setchapternewpage`. Команда `@headings` нужна вам только в том случае, если команда `@setchapternewpage` делает не то, что вам нужно, или если вы хотите выключить предопределенные заголовки страниц до определения ваших собственных. Пишите команду `@headings` сразу после команды `@end titlepage`.

Вы можете использовать `@headings` так:

`@headings off`
: Выключает печать заголовков страниц.

`@headings single`
: Включает заголовки страниц, подходящие для односторонней печати.

`@headings double`
`@headings on`
: Включает заголовки страниц, подходящие для двусторонней печати. Команды `@headings on` и `@headings double` являются синонимами.

`@headings singleafter`
`@headings doubleafter`
: Включает заголовки `single` или `double`, соответственно, после текущей выводимой страницы.

`@headings on`
: Включает заголовки страниц: `single`, если '`@setchapternewpage on`', и `double` в противном случае.

Например, предположим, что вы вставили команду `@setchapternewpage off` перед командой `@titlepage`, чтобы заставить TEX начинать новую главу на той же самой странице, где закончилась предыдущая. Эта команда также заставляет TEX набирать заголовки страниц для односторонней печати. Чтобы TEX форматировал для двусторонней печати, напишите после `@end titlepage` команду `@headings double`.

Вы можете совсем запретить формирование заголовков страниц путем вставки команды `@headings off` на отдельной строке сразу после строки с командой `@end titlepage`, как показано:

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

Команда `@headings off` перекрывает действие команды `@end titlepage`, которая в противном случае заставила бы TEX печатать заголовки страниц.

Также, вы можете задать собственный стиль верхних и нижних заголовков. См. Приложение F [Заголовки страниц], с. 206, для более подробной информации.

## 3.5 Нода Top и главное меню

Нода 'Top' — это нода, с которой вы входите в Info-файл.

Нода 'Top' должна содержать краткое описание Info-файла и большое главное меню для доступа ко всему Info-файлу. Это помогает читателю понять, какая информация содержится в данном Info-файле. Кроме того, вы должны указать версию программы, к которой применим этот Info-файл, или по меньшей мере номер редакции Info-файла.

Содержимое ноды 'Top' должно появляться только в Info-файле и не должно печататься, поэтому заключайте его между команд `@ifinfo` и `@end ifinfo`. (TEX не печатает ни строки `@node`, ни меню; они появляются только в Info-файле. То есть, строго говоря, вам не нужно заключать эти части их между команд `@ifinfo` и `@end ifinfo`, но проще всего все же сделать это. См. Глава 16 [Условно видимый текст], с. 134.)

### 3.5.1 Заголовок в ноде Top

Иногда вам нужно будет вставить команду структурирования `@top`, содержащую название документа, сразу после строки `@node Top` (См. Раздел 5.3 [Команда @top], с. 47, для более детальной информации).

Например, нода 'Top' данного руководства содержит в начале команду описания структуры глав `@top`, краткое описание, номер редакции документа и версию программы. И все это выглядит следующим образом:

```
...
@end titlepage

@ifnottex
@node Top, Копирование, , (dir)
@top Texinfo
Texinfo -- это система документации...
Это версия ...
...
@end ifnottex

@menu
* Копирование::                 Ваши права.
* Обзор::                       Коротко о Texinfo.
...
@end menu
```

Указатели ‘Previous’ и ‘Up’ в ноде ‘Top’ обычно ссылаются на каталог верхнего уровня всей системы Info, который называется ‘`(dir)`’. Нода ‘Next’ ссылается на первую ноду, которая следует сразу за главным меню; обычно это разрешение на копирование, введение или первая глава.

### 3.5.2 Части главного меню

*Главное меню* — это подробное меню, содержащее ссылки на все ноды в файле.

Главное меню заключается между команд `@menu` и `@end menu` и не появляется в печатном руководстве.

Как правило, главное меню делится на части.

- Первая часть содержит основные ноды Texinfo-файла: ноды для глав, разделов, подобных главам, и для приложений.
- Вторая часть содержит ноды для именных указателей.
- Третья и последующие части содержат полный список нод более низких уровней, обычно упорядоченный по главам. Через него читатель может перейти непосредственно к конкретной ноде при поиске определенных сведений, не проходя через промежуточные меню. Эти пункты меню не являются необходимыми; добавьте их, если это послужит удобству пользования. Если вы решили их использовать, напишите команду `@detailmenu` перед первым пунктом и `@end detailmenu` после последнего, иначе в работе `makeinfo` возникнут проблемы.

Каждая часть меню может быть представлена строкой описания. Если строка не начинается со звездочки, она не будет восприниматься как пункт меню. (См. Раздел 7.1 [Написание меню], с. 61, для более детальной информации.)

Например, главное меню этого руководства выглядит следующим образом (но в нем гораздо больше пунктов):

```
@menu
* Копирование::                 Ваши права.
* Обзор::                       Коротко о Texinfo.
* Режим Texinfo::               Как использовать режим Texinfo.
...
...
* Указатель команд и переменных::
* Указатель понятий::           Меню, охватывающее все темы.

@detailmenu
 -- Полный список нод --

Обзор Texinfo

* Описание ошибок::             Посылка эффективных сообщений об
                                ошибках.
* Использование Texinfo::       Создание обычной печатной книги или
                                Info-файла.
...
...

Использование режима Texinfo
* Обзор режима Texinfo::        Как режим Texinfo может вам помочь.
...
...
@end detailmenu
@end menu
```

## 3.6 Разрешение на копирование программы

Если Texinfo-файл имеет раздел, содержащий “Универсальную Общественную Лицензию”, разрешение на распространение и отказ от предоставления гарантий для документируемого программного обеспечения, этот раздел обычно следует после ноды ‘Top’. Универсальная Общественная Лицензия — это юридический документ, который очень важен для программного обеспечения Проекта GNU. Он дает вам и другим людям уверенность в том, что право на использование и распространение данного программного обеспечения у вас не отнимут.

После информации о копировании и распространении и отказа от предоставления гарантий следует введение или первая глава руководства.

Хотя введение не является обязательной частью Texinfo-файла, присутствие его в документе очень полезно. В идеальном случае во введении должно быть кратко и ясно изложено, о чем рассказывается в данном Texinfo-файле и кому он может быть интересен. Вообще говоря, введение следует после информации о разрешениях на распространение, хотя иногда некоторые люди помещают эту часть в документе раньше. Обычно введение помещается в ненумерованном разделе (созданном командой `@unnumbered`). (См. Раздел 5.5 [Команды `@unnumbered` и `@appendix`], с. 48.)

# 4 Завершение Texinfo-файла

В конце Texinfo-файла должны быть команды для создания именных указателей и (обычно) содержаний, подробного и обзорного. Он также должен включать команду `@bye`, помечающую последнюю обрабатываемую TEX строку.

Например:

```
@node    Указатель понятий, ,Указатель переменных, Top
@c       имя-ноды, следующая, предыдущая, вверх
@unnumbered Указатель понятий

@printindex cp

@contents
@bye
```

## 4.1 Меню-указатели и печать именных указателей

Напечатать именной указатель — значит включить его в руководство или Info-файл. Это не делается автоматически, потому что в Texinfo-файле вы используете команду `@cindex` или другие команды, создающие вхождения именного указателя; они только собирают исходные данные для создания указателя. Чтобы создать именной указатель, вы должны поместить в том месте документа, где вы хотите его увидеть, команду `@printindex`. Кроме этого, при создании печатного руководства вы должны запустить программу, называемую `texindex` (см. Глава 19 [Печать], с. 146), которая сортирует исходные данные и записывает сортированный именной указатель в файл. Именно этот файл используется для печати именного указателя.

В Texinfo предопределены шесть различных типов именных указателей: указатель понятий, указатель функций, указатель переменных, указатель клавиш, указатель программ и указатель типов данных (см. Раздел 12.2 [Предопределенные именные указатели], с. 100). Каждый указатель имеет двухбуквенное имя: ‘`cp`’, ‘`fn`’, ‘`vr`’, ‘`ky`’, ‘`pg`’ и ‘`tp`’. Вы можете объединить указатели или поместить их в разные разделы (см. Раздел 12.4 [Объединение именных указателей], с. 102); вы также можете определить свои именные указатели (см. Раздел 12.5 [Новые именные указатели], с. 104).

Команда `@printindex` принимает в качестве аргумента двухбуквенное имя именного указателя, читает соответствующий файл с сортированным указателем и форматирует указатель соответствующим образом.

Команда `@printindex` не создает заголовка главы для именного указателя. Следовательно, перед командой `@printindex` вы должны написать подходящую команду создания главы или раздела (обычно `@unnumbered`), чтобы указатель имел заголовок и был упомянут в содержании. Перед командой `@unnumbered` пишите строку `@node`.

Например:

```
@node Указатель переменных, Указатель понятий, Указатель функций, Top
@comment имя-ноды, следующая, предыдущая, вверх
@unnumbered Указатель переменных

@printindex vr
```

```
@node     Указатель понятий,     , Указатель переменных, Top
@comment имя-ноды, следующая, предыдущая, вверх
@unnumbered Указатель понятий

@printindex cp
```

Читателям нравится, когда указатель понятий расположен последним в книге, потому что тогда его легче найти. Наличие только одного указателя также помогает читателям, так как тогда них у есть только одно место для поиска (см. Раздел 12.4.2 [synindex], с. 104)

## 4.2 Создание содержания

Команды `@chapter`, `@section` и другие команды структурирования предоставляют необходимые сведения для создания содержания, но не приводят к действительному появлению содержания в руководстве. Чтобы сделать его, вы должны использовать команды `@contents` и/или `@summarycontents`.

`@contents`
: Создает содержание печатного руководства, включающее главы, разделы, подразделы и так далее, а также приложения и ненумерованные главы. (Заголовки, созданные последовательностями команд `@heading`, не появляются в содержании.)

`@shortcontents`
`@summarycontents`
: (`@summarycontents` — синоним `@shortcontents`; две эти команды совершенно одинаковы.)

  Создают короткое или, иначе говоря, обзорное содержание, в котором перечисляются только главы (и приложения, и ненумерованные главы). Разделы, подразделы и подподразделы пропускаются. Короткое содержание требуется только для больших руководств, в дополнение к полному содержанию.

Обе команды для содержаний нужно писать на отдельной строке. Команды для содержаний автоматически создают заголовок в начале первой страницы содержания, поэтому не включайте перед ними никаких команд для описания структуры глав, таких как `@unnumbered`.

Так как в Info-файлах вместо содержания используются меню, команды форматирования для Info игнорируют команды для вывода содержаний. Но содержания включаются в вывод в простом текстовом формате (создаваемом командой `makeinfo --no-headers`).

Команды печати содержания можно помещать как в самом конце файла после всех именных указателей (смотрите предыдущий раздел), непосредственно перед командой `@bye` (смотрите следующий раздел), или недалеко от начала файла, после `@end titlepage` (см. Раздел 3.4.1 [titlepage], с. 35). Преимущество первого способа в том, что вывод содержания всегда соответствует истине, потому что он отражает только что сделанные изменения. Преимущество второго в том, что содержание печатается

в правильном месте, так что вам не нужно переорганизовывать DVI-файл или перетасовывать бумагу. Однако команды для печати содержания в начале документа игнорируются при выводе на стандартный вывод.

Как автор, вы можете поместить команды для печати содержания где вы предпочитаете. Но если вы просто печатаете руководство, вы можете пожелать напечатать содержание после титульного листа даже если автор поместил команды для печати содержания в конец документа (как в случае большинства существующих документов Texinfo). Вы можете сделать это задав `@setcontentsaftertitlepage` и/или `@setshortcontentsaftertitlepage`. Первое печатает только главное содержание после `@end titlepage`; второе печатает и краткое, и полное содержание. В любом случае любые последующие команды `@contents` или `@shortcontents` игнорируются (если только `@end titlepage` не была встречена вообще).

Вам нужно включать команды `@set...contentsaftertitlepage` в начале документа (сразу после `@setfilename`, например). Или, если вы пользуетесь `texi2dvi` (см. Раздел 19.3 [Форматирование с texi2dvi], с. 148), вы можете использовать его ключ '`-texinfo`' для задания этого вообще без изменения исходного файла. Например:

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

## 4.3 `@bye` Завершение файла

Команда `@bye` прекращает форматирование в TEX или Info. Все, что следует в файле после команды `@bye`, невидимо для форматирующих команд. Команда `@bye` должна находиться на отдельной строке.

Вы можете, если хотите, писать заметки после строки `@bye`. Эти заметки не будут форматироваться и не появятся ни в Info-файле, ни в печатном руководстве, как если бы текст после `@bye` был помещен внутри блока `@ignore ... @end ignore`. Вы также можете написать после строки `@bye` список локальных переменных. См. Раздел 19.7 [Использование локальных переменных и команда компилирования], с. 151, для подробных сведений.

# 5 Структура глав

Команды описания *структуры глав* разделяют документ на иерархию глав, разделов, подразделов и подподразделов. Эти команды создают большие заголовки, а также предоставляют сведения для составления содержания печатного руководства (см. Раздел 4.2 [Создание содержания], с. 44).

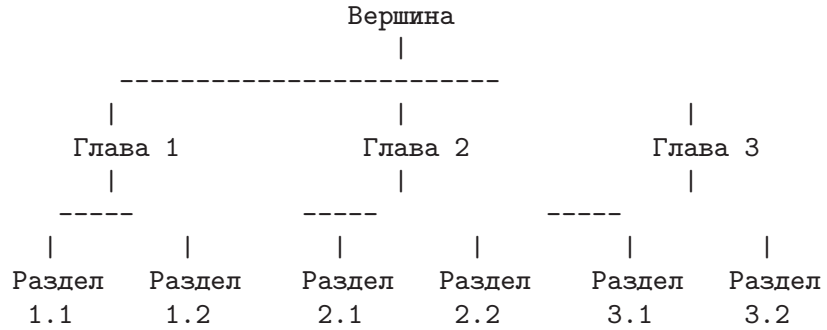
Команды описания структуры глав не создают структуры Info-нод, поэтому обычно вы должны помещать команду `@node` непосредственно перед каждой командой описания структуры глав (см. Глава 6 [Ноды], с. 53). Единственный случай, когда вы, вероятно, будете использовать команды описания структуры глав, не описывая структуру нод, — это если вы пишите документ, не содержащий перекрестных ссылок, и который никогда не будет преобразован в формат Info.

Вряд ли вам когда-нибудь придется писать Texinfo-файл, предназначенный только для чтения в виде Info-файла, но не печатного документа. Если же вам пришлось, все равно имеет смысл использовать команды описания структуры глав для создания заголовка в начале каждой ноды, но вам не обязательно это делать.

## 5.1 Древовидная структура разделов

Обычно Texinfo-файл организован подобно книге с главами, разделами, подразделами и так далее. Такую структуру можно изобразить в виде дерева (или даже дерева, направленного сверху вниз) с корнем наверху и уровнями, соответствующими главам, разделам, подразделам и подподразделам.

Ниже приведена диаграмма, изображающая Texinfo-файл из трех глав, каждая из которых имеет два раздела.




В Texinfo-файле с такой структурой начало Главы 2 выглядит таким образом:

```
@node    Глава 2,  Глава 3, Глава 1, top
@chapter Глава 2
```

Команды описания структуры глав рассмотрены в последующих разделах; команды `@node` и `@menu` — в последующих главах. (См. Глава 6 [Ноды], с. 53, и Глава 7 [Меню], с. 61.)

## 5.2 Типы команд описания структуры

Команды описания структуры глав делятся на четыре группы или серии, каждая из которых содержит команды описания структур, соответствующих иерархическому уровню глав, разделов, подразделов и подподразделов.

Эти четыре группы составляют: серия `@chapter`, серия `@unnumbered`, серия `@appendix` и серия `@heading`.

Каждая команда создает заголовки, выглядящие по-разному на напечатанной странице и в Info-файле; только некоторые команды создают заголовки, которые будут перечислены в содержании печатной книги или руководства.

- Серия команд `@chapter` и `@appendix` создает нумерованные или перечисленные по буквам вхождения как в теле печатного произведения, так и в его содержании.
- Серия команд `@unnumbered` создает ненумерованные вхождения как в теле печатного произведения, так и в его содержании. Команда `@top`, имеющая особое значение, является членом этой серии (см. Раздел 5.3 [@top], с. 47).
- Серия команд `@heading` создает ненумерованные заголовки, которые не появляются в содержании. Команды создания заголовка никогда не начинают новую страницу.
- Команда `@majorheading` дает результат, похожий на результат применения команды `@chapheading`, но делает больший вертикальный пропуск перед заголовком.
- Если так установлено с помощью команды `@setchapternewpage`, то `@chapter`, `@unnumbered`, и `@appendix` начинают новую страницу в печатном руководстве; команда `@heading` не начинает.

Вот четыре группы команд описания структуры глав:

| | | | Не начинают новую страницу |
|---|---|---|---|
| Нумерованные | Ненумерованные | Нумерованные или перечисленные по буквам | Ненумерованные |
| В содержании | В содержании | В содержании | Нет в содержании |
| | `@top` | | `@majorheading` |
| `@chapter` | `@unnumbered` | `@appendix` | `@chapheading` |
| `@section` | `@unnumberedsec` | `@appendixsec` | `@heading` |
| `@subsection` | `@unnumberedsubsec` | `@appendixsubsec` | `@subheading` |
| `@subsubsection` | `@unnumberedsubsubsec` | `@appendixsubsubsec` | `@subsubheading` |

## 5.3 `@top`

Команда `@top` — это специальная команда, которую вы используете после строки ‘`@node Top`’ в начале Texinfo-файла. Команда `@top` сообщает форматирующей програм-

ме `makeinfo`, какая нода является нодой ‘Top’, чтобы `makeinfo` могла использовать эту ноду в качестве корневой, если в вашем руководстве применяются неявные указатели. Ее результат при наборе такой же, как при применении `@unnumbered` (см. Раздел 5.5 [@unnumbered и @appendix], с. 48). Для подробной информации смотрите Раздел 6.3.6 [Команда @top], с. 58.

Нода `@top` и ее меню (если оно есть) обычно окружается в условную конструкцию `@ifnottex`, чтобы они появлялись только при выводе в выводе Info и HTML, но не TEX.

## 5.4 @chapter

Команда `@chapter` обозначает главу в документе. Пишите эту команду в начале строки и за ней название главы на той же строке.

Например, данная глава в этом руководстве озаглавлена “Структура глав”; строка `@chapter` выглядит так:

```
@chapter Структура глав
```

В TEX, команда `@chapter` создает в документе главу и задает название этой главы. Глава автоматически получает номер.

В Info, команда `@chapter` создает название на отдельной строке и строку звездочек под ним. Таким образом, в Info предыдущий пример дает такой вывод:

```
Структура глав
**************
```

Texinfo также предоставляет команду `@centerchap`; она аналогична `@unnumbered`, но центрирует свой аргумент в печатном выводе. Обычно такой стиль не рекомендуется к использованию в Texinfo.

## 5.5 @unnumbered и @appendix

Используйте команду `@unnumbered` для создания в печатном руководстве главы без какого-либо номера. Используйте команду `@appendix` для создания в печатном руководстве приложения, помеченного буквой, а не числом.

При выводе в Info-файл команды `@unnumbered` и `@appendix` действуют эквивалентно `@chapter`: заголовок печатается на отдельной строке, а под ним печатается строка звездочек. (См. Раздел 5.4 [@chapter], с. 48.)

Чтобы создать приложение или ненумерованную главу, напишите в начале строки команду `@appendix` или `@unnumbered`, а после нее на той же строке — название, как бы вы написали при создании главы.

## 5.6 @majorheading, @chapheading

Команды `@majorheading` и `@chapheading` помещают в тело документа заголовки, подобные заголовкам глав.

Однако, эти команды не говорят TEX создавать нумерованные заголовки или вхождения в содержании, а также начинать в печатном руководстве новую страницу.

В TEX, команда `@majorheading` создает больший вертикальный пропуск перед заголовком, чем команда `@chapheading`, но в остальном эти команды одинаковы.

В Info, команды `@majorheading` и `@chapheading` эквивалентны `@chapter`: заголовок печатается на отдельной строке, и под ним печатается строка звездочек. (См. Раздел 5.4 [@chapter], с. 48.)

## 5.7 @section

В печатном руководстве, команда `@section` обозначает нумерованный раздел внутри главы. Заголовок раздела появляется в содержании. В Info команда `@section` делает для сегмента текста заголовок, подчеркнутый символами '='.

Данный раздел озаглавлен с помощью команды `@section` следующим образом:

```
@section @code{@@section}
```

Чтобы создать раздел, напишите команду `@section` в начале строки, и после нее название раздела на той же строке.

Таким образом,

```
@section Это раздел
```

дает в Info

```
Это раздел
==========
```

## 5.8 @unnumberedsec, @appendixsec, @heading

Команды `@unnumberedsec`, `@appendixsec` и `@heading` — это эквиваленты команды `@section` для, соответственно, ненумерованных разделов, приложений и заголовков. (См. Раздел 5.7 [@section], с. 49.)

`@unnumberedsec`
: Команда `@unnumberedsec` может применяться внутри ненумерованной или обычной главы или приложения для создания ненумерованного раздела.

`@appendixsec`
`@appendixsection`
: `@appendixsection` — более длинный вариант написания `@appendixsec`; эти команды являются синонимами.

  Команда `@appendixsec` или `@appendixsection` по соглашению используется только внутри приложений.

`@heading`
: Вы можете использовать команду `@heading` везде, где захотите, для создания заголовка, подобного заголовку раздела, который не появится в содержании.

## 5.9 Команда `@subsection`

Подразделы соотносятся с разделами так же, как разделы с главами. (См. Раздел 5.7 [`@section`], с. 49.) В Info заголовки подразделов подчеркиваются символами '-'. Например,

```
@subsection Это подраздел
```

дает

```
Это подраздел
---------
```

В печатном руководстве подразделы вносятся в содержание и нумеруются до третьего уровня глубины.

## 5.10 Команды, подобные `@subsection`

Команды `@unnumberedsubsec`, `@appendixsubsec` и `@subheading` — это эквиваленты команды `@subsection` для, соответственно, ненумерованных разделов, приложений и заголовков. (См. Раздел 5.9 [`@subsection`], с. 50.)

В Info команды, подобные `@subsection`, создают заголовок, подчеркнутый строкой дефисов. В печатном руководстве, команда `@subheading` создает заголовок, как для подраздела, за исключением того, что он не нумеруется и не вносится в содержание. Аналогично, команда `@unnumberedsubsec` создает ненумерованный заголовок, как для подраздела, а команда `@appendixsubsec` — заголовок, подобный заголовку подраздела, помеченный буквой и числами; заголовки, создаваемые обеими этими командами, вносятся в содержание.

## 5.11 Команды subsub

Команды Texinfo для описания структуры глав четвертого, нижнего уровня — это команды, начинающиеся с 'subsub'. Они включают:

`@subsubsection`
: Подподразделы соотносятся с подразделами так же, как подразделы с разделами. (См. Раздел 5.9 [`@subsection`], с. 50.) В печатном руководстве заголовки подподразделов вносятся в содержание и нумеруются до четвертого уровня.

`@unnumberedsubsubsec`
: Заголовки ненумерованных подподразделов вносятся в содержание печатного руководства, но не имеют номеров. В остальном ненумерованные подподразделы эквиваленты простым подподразделам. В Info ненумерованные подподразделы выглядят точно так же, как и простые подподразделы.

`@appendixsubsubsec`
: По соглашению, команды для приложений используются только внутри приложений. В печатном руководстве они создают заголовки, помеченные буквами или числами. Эти заголовки также появляются в содержании. В Info, подподразделы приложения выглядят так же, как и простые подподразделы.

`@subsubheading`

Команда `@subsubheading` применяется везде, где вам нужен маленький заголовок, который не вносится в содержание. В Info такие заголовки выглядят так же, как и заголовки простых подподразделов.

В Info заголовки подподразделов подчеркиваются строкой точек. Например,

```
@subsubsection Это подподраздел
```

дает

```
Это подподраздел
.................
```

## 5.12 `@raisesections` и `@lowersections`

Команды `@raisesections` и `@lowersections` повышают и понижают иерархический уровень глав, разделов, подразделов и подобных сегментов. Команда `@raisesections` меняет разделы на главы, подразделы на разделы и так далее. Команда `@lowersections` меняет главы на разделы, разделы на подразделы и так далее.

Команда `@lowersections` полезна, если вы хотите включить текст, записанный в отдельном или самостоятельном Texinfo-файле, в виде внутреннего, включаемого файла. Если вы напишите эту команду в начале файла, все команды `@chapter` будут отформатированы, как если бы они были командами `@section`, все команды `@section` будут отформатированы как команды `@subsection`, и так далее.

`@raisesections` увеличивает уровень команды в иерархии структурирования глав на один:

| Изменяет | на |
|---|---|
| `@subsection` | `@section`, |
| `@section` | `@chapter`, |
| `@heading` | `@chapheading`, |
| etc. | |

`@lowersections` уменьшает уровень команды в иерархии структурирования глав на один:

| Изменяет | на |
|---|---|
| `@chapter` | `@section`, |
| `@subsection` | `@subsubsection`, |
| `@heading` | `@subheading`, |
| etc. | |

Команда `@raisesections` или `@lowersections` изменяет только те команды описания структуры, которые идут в исходном Texinfo-файле после нее. Пишите команды `@raisesections` и `@lowersections` на отдельной строке.

Команда `@lowersections` отменяет действие `@raisesections` и наоборот. Как правило, эти команды используются следующим образом:

```
@lowersections
@include somefile.texi
```

`@raisesections`

Если не написать `@raisesections`, все последующие разделы в вашем документе будут на уровень ниже.

Повторное применение этих команд продолжает поднимание или опускание иерархического уровня, на один каждый раз.

Попытка подняться выше уровня глав дает снова команды для глав, попытка спуститься ниже уровня подподразделов дает снова команды для подподразделов.

# 6 Ноды

*Ноды* — это первичные сегменты Texinfo-файла. Сами по себе они не налагают иерархической или иной другой структуры на файл. Ноды содержат *указатели на ноды*, которые отсылают к другим нодам, и *меню*, являющиеся списками нод. Команды перемещения в Info могут привести вас к ноде по указателю на нее или к ноде, перечисленной в меню. Указатели на ноды и меню обеспечивают некую структуру в Info-файлах, как главы, разделы, подразделы и подобные фрагменты обеспечивают структуру в печатных книгах.

## 6.1 Два способа

Команды для нод и меню и команды описания структуры глав независимы друг от друга в техническом смысле:

- В Info структура обеспечивается командами для нод и меню. Команды описания структуры глав создают заголовки с различными видами подчеркивания — звездочками для глав, дефисами для разделов и так далее; они не делают ничего больше.
- В TEX команды описания структуры глав генерируют содержание и номера глав и разделов. Команды для нод и меню предоставляют сведения для создания перекрестных ссылок; они не делают ничего больше.

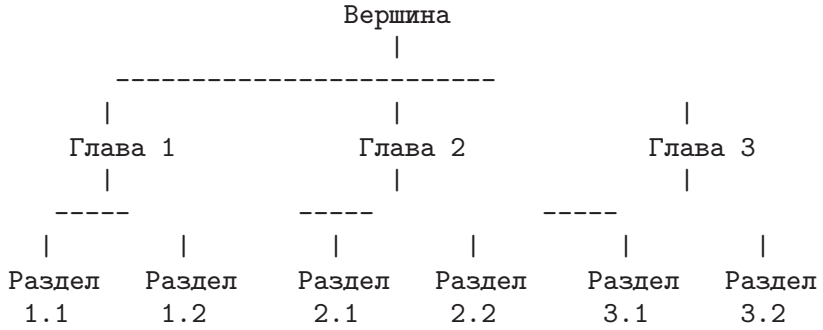
Вы можете использовать указатели на ноды и меню, чтобы создать Info-файл с любой структурой, какой захотите; вы можете написать Texinfo-файл так, что его вывод в формате Info будет иметь другую структуру, нежели его печатный вывод. Однако, прктически все Texinfo-файлы наисаны так, что структура вывода в Info соответствует структуре печатного вывода. Делать иначе было бы неудобно и непонятно для читателя.

Обычно печатный вывод структурирован в виде древовидной иерархии, в которой главы являются более крупными членами, и от них ответвляются разделы. Аналогично, указатели на ноды и меню организуются так, чтобы создать совпадающую структуру при выводе в Info.

## 6.2 Иллюстрация нод и меню

Здесь приведена копия показанной ранее диаграммы, которая изображает Texinfo-файл с тремя главами, каждая из которых содержит три раздела.

“Корень” находится в вершине диаграммы, а “листья” внизу. Традиционно подобные диаграммы изображаются именно таким образом; они иллюстрируют дерево, направленное сверху вниз. По этой причине корневая нода называется ‘Top’ (Первая), а указатели ‘Up’ (Вверх) переносят вас ближе к корню.




Полная форма команды для начала Главы 2 выглядела бы так:

```
@node    Глава 2,   Глава 3, Глава 1,  Top
@comment node-name, next,    previous, up
```

Эта строка `@node` говорит, что данная нода называется “Глава 2”, следующая нода называется “Глава 3”, предыдущая нода называется “Глава 1”, а верхняя нода называется “Top”. Вы можете не писать все эти имена нод, если ваш документ организован иерархически (см. Раздел 6.4 [Создание указателей с makeinfo], с. 59), но связь между указателями сохраняется.

> **Пожалуйста, обратите внимание:** ‘Next’ ссылается на следующую ноду на том же иерархическом уровне руководства, не обязательно на следующую ноду в Texinfo-файле. В Texinfo-файле, последующие ноды могут находиться на более нижнем уровне: ноды уровня разделов чаще всего следуют за нодами уровня глав, для примера. Указатели ‘Next’ и ‘Previous’ ссылаются на ноды *того же* иерархического уровня. (Нода ‘Top’ представляет исключение из этого правила. Так как нода ‘Top’ является единственной нодой на своем уровне, то ее указатель ‘Next’ ссылается на первую последующую ноду, которая почти всегда является главой или нодой уровня глав.)

Чтобы перейти к разделам 2.1 и 2.2, используя Info, вам понадобится меню внутри Главы 2. (См. Глава 7 [Меню], с. 61.) Вы должны написать меню непосредственно перед началом Раздела 2.1, как показано ниже:

```
@menu
* Разд. 2.1::    Описание этого раздела.
* Разд. 2.2::
@end menu
```

Напишите ноду для раздела 2.1 так:

```
@node     Разд. 2.1, Разд. 2.2, Глава 2, Глава 2
@comment  node-name, next,      previous,  up
```

В формате Info указатели ‘Next’ и ‘Previous’ обычно ведут к другим нодам того же уровня — от главы к главе или от раздела к разделу (иногда, как показано, ‘Previous’ ссылается на верхнюю ноду); указатель ‘Up’, как правило, ведет к ноде верхнего уровня (ближе к первой ноде (‘Top’); меню приводит к нодам более низкого уровня (ближе к ‘листьям’). (Перекрестная ссылка может указывать на ноду любого уровня; смотрите Глава 8 [Перекрестные ссылки], с. 65.)

Обычно команда `@node` и команды описания структуры глав используются вместе, как и команды добавления вхождений в именные указатели. (Вы можете написать после строки `@node` строку комментария, который напомнит вам, какой указатель куда указывает.)

Ниже приведено начало главы данного руководства, озаглавленной “Завершение Texinfo-файла”. Тут показана строка `@node`, за которой идет строка комментария, строка `@chapter` и затем строки для именных указателей.

```
@node Завершение файла, Структурирование, Начало файла, Top
@comment node-name, next, previous, up
@chapter Завершение Texinfo-файла
@cindex Завершение Texinfo-файла
@cindex Texinfo-файл, завершение
@cindex Файл, завершение
```

## 6.3 Команда `@node`

*Нода* — это сегмент текста, начинающийся с команды `@node` и продолжающийся до следующей команды `@node`. Определение ноды отличается от определения главы или раздела. Глава может включать разделы, а раздел — подразделы, но нода не может содержать подноды; текст ноды продолжается только до следующей команды `@node` в файле. Нода обычно содержит только одну команду для описания структуры глав, ту, что идет после строки `@node`. С другой стороны, в печатном выводе ноды применяются только для перекрестных ссылок, так что глава или раздел может содержать любое число нод. На самом деле, глава обычно включает несколько нод, по одной на каждый раздел, подраздел и подподраздел.

Чтобы создать ноду, напишите команду `@node` в начале строки и за ней четыре ее аргумента, разделенные запятыми, в конце той же строки. Первый аргумент обязателен: это имя данной ноды. Последующие аргументы — это имена для указателей ‘Next’, ‘Previous’ и ‘Up’, в таком порядке; они могут быть опущены, если ваш документ организован иерархически (см. Раздел 6.4 [Создание указателей с makeinfo], с. 59).

Вы можете вставлять пробелы перед каждым именем, если хотите. Вы должны писать имя ноды и имена указателей ‘Next’, ‘Previous’ и ‘Up’ на одной строке, иначе программы форматирования не смогут их правильно обработать. (См. Info файл ‘`info`’, node ‘`Top`’, для получения подробной информации о нодах в Info.)

Обычно сразу после строки `@node` вы будете писать строку с одной из команд описания структуры глав — например, строку `@section` или `@subsection`. (См. Раздел 5.2 [Типы команд структурирования], с. 47.)

> **Пожалуйста, обратите внимание:** Команды обновления в режиме Texinfo для GNU Emacs работают только с такими Texinfo-файлами, в которых после строк `@node` написаны строки описания структуры глав. См. Раздел 2.4.1 [Требования для обновления], с. 21.

TEX использует строки `@node` для определения имен перекрестных ссылок. Поэтому вы должны писать строки `@node` в Texinfo-файле, предназначенном для форматирования для печати, даже если вы не намереваетесь форматировать его для Info. (Перекрестные ссылки, такие как в конце этого предложения, создаются командой `@xref` и родственными с ней командами; смотрите Глава 8 [Перекрестные ссылки], с. 65.)

### 6.3.1 Выбор имен нод и указателей

Имя ноды служит для ее идентификации. Указатели позволяют вам достичь других нод и состоят из имен этих нод.

Как правило, указатель 'Up' содержит имя ноды, в меню которой упомянута данная нода. Указатель 'Next' содержит имя ноды, следующей после данной в этом меню, а указатель 'Previous' — имя ноды, написанной в меню перед данной. Когда ноды 'Previous' и 'Up' совпадают, оба указателя ссылаются на одну ноду.

Обычно первой нодой Texinfo-файла является нода 'Top', а ее указатели 'Up' и 'Previous' ссылаются на файл '`dir`', который содержит главное меню для всей системы Info.

Сама нода 'Top' содержит главное или мастер-меню руководства. Также в ноду 'Top' полезно включить краткое описание этого руководства. См. Раздел 6.3.5 [Первая нода], с. 58, чтобы узнать, как писать первую ноду Texinfo-файла.

Даже если вы явно записываете все указатели, это не означает, что вы можете писать ноды в исходном Texinfo-файле в произвольном порядке! Так как TEX обрабатывает файл последовательно, не обращая внимания на указатели на ноды, вы должны писать ноды в том порядке, в каком вы желаете их видеть в печатном выводе.

### 6.3.2 Как писать строку `@node`

Простейший способ написать строку `@node` — написать в начале строки команду `@node` и затем имя ноды, как показано здесь:

```
@node имя-ноды
```

Если вы пользуетесь GNU Emacs, вы можете использовать для вставки имен указателей команды обновления ноды, предоставляемые режимом Texinfo; или вы можете не заботиться об указателях в Texinfo-файле и предоставить программе `makeinfo` вставить их в создаваемый ей Info-файл. (См. Глава 2 [Режим Texinfo], с. 15, и Раздел 6.4 [Создание указателей с makeinfo], с. 59.)

Или вы можете вставить указатели 'Next', 'Previous' и 'Up' сами. Если вы делаете так, вы можете счесть полезным использовать команду режима Texinfo `C-c C-c n`. Эта команда вставляет строку '`@node`' и строку комментария, перечисляющего имена указателей на ноды в нужном порядке. Такая строка комментария особенно полезна, если вы не совсем освоились с Texinfo.

Шаблон для полной формы строки ноды с указателями 'Next', 'Previous' и 'Up' выглядит следующим образом:

```
@node имя-ноды, следующая, предыдущая, вверх
```

Если вы хотите, то можете вообще не писать строки `@node` в первом наброске и затем использовать команду `texinfo-insert-node-lines`, которая создаст их за вас. Однако мы не рекомендуем такую практику. Лучше давать ноде имя тогда же, когда вы пишите ее текст, чтобы вы могли легко создавать перекрестные ссылки. Большое число перекрестных ссылок — это особенно важная отличительная черта хорошего Info-файла.

После того, как вы вставили строку `@node`, вы должны сразу написать `@`-команду для главы или раздела и вставить имя этой главы. Потом (и это важно!) напишите

несколько вхождений для именных указателей. Обычно вы сможете найти по крайней мере два, а часто даже четыре или пять способов сослаться на ноду из именного указателя. Используйте их все. Это позволит людям намного легче найти данную ноду.

### 6.3.3 Советы по написанию строки `@node`

Вот три рекомендации:

- Старайтесь подбирать для нод короткие, но информативные имена.

  В Info-файле, имена файла, ноды и указателей вставляются в одну строку, которая может уйти за правый край окна. (Это не вызывает неприятностей в Info, но смотрится плохо.)
- Старайтесь подбирать для нод имена с различными первыми буквами. Так будет легче использовать автоматическое завершение имен в Info.
- По соглашению, в именах нод заглавные буквы используются так же, как в заголовках глав и разделов — первое слово и значимые слова пишутся с заглавной буквы, остальные — со строчной.[1]

### 6.3.4 Требования для строки `@node`

Вот несколько требований, предъявляемых к строкам `@node`:

- Все имена нод в одном Info-файле должны быть уникальны.

  Дублирования вводят в заблуждение команды перемещения Info. Это значит, например, что если вы завершаете каждую главу обзором, вы должны называть каждую ноду с обзором по-разному. Вы не можете просто назвать их все “Обзор”. Вы можете, однако, дублировать названия глав, разделов и подобных сегментов. Таким образом, вы можете завершать каждую главу разделом, озаглавленным “Обзор”, пока имена нод для всех этих разделов различны.
- Имя указателя должно быть именем ноды.

  Нода, на которую ссылается указатель, может идти до или после ноды, содержащей этот указатель.
- `@`-команды, использованные в именах нод, обычно вводят в заблуждение Info, поэтому вам лучше избегать их. Для тех редких случаев, когда это бывает полезно, в Texinfo есть ограниченная поддержка использования `@`-команд в именах нод; смотрите Раздел 20.1.4 [Проверка указателей], с. 162.

  Вы не можете использовать какие-либо `@`-команды Texinfo в имени ноды, они вводят в заблуждение Info.

  Таким образом, начало раздела, называемого `@chapter` выглядит следующим образом:

```
@node chapter, unnumbered & appendix, makeinfo top, Структурирование
@section @code{@@chapter}
@findex chapter
```

[1] В русских текстах с заглавной буквы пишется только первое слово. *(Прим. переводчика)*

- К сожалению, вы не можете использовать точки, запятые, двоеточия или апострофы в имени ноды; эти символы вводят в заблуждение TEX или программы форматирования для Info.

  Например, ниже приведен заголовок раздела:

  ```
  @code{@@unnumberedsec}, @code{@@appendixsec},
  @code{@@heading}
  ```

  Имя соответствующей ноды:

  ```
  unnumberedsec appendixsec heading
  ```

- Регистр имеет значение.

### 6.3.5 Первая нода

Первой нодой Texinfo-файлой, за исключением включаемых файлов (см. Приложение E [Включаемые файлы], с. 202), является нода *Top*. Первая нода содержит главное меню документа и его краткий обзор. (см. Раздел 6.3.7 [Обзор в первой ноде], с. 59).

Первая нода (которая должна называться ‘`top`’ или ‘`Top`’) должна содержать в качестве указателя ‘Up’ имя ноды в другом файле, в котором есть меню, ведущее к данному файлу. Имя файла пишется в круглых скобках. Если файл должен быть установлен непосредственно в файл-каталог Info, пишите в качестве родителя первой ноды ‘`(dir)`’; это сокращение для ‘`(dir)top`’, указывающее на первую ноду в файле ‘`dir`’, которая содержит главное меню для всей системы Info. Например, в этом руководстве строка `@node Top` выглядит так:

```
@node Top, Копирование, , (dir)
```

(Вы можете использовать команды обновления Texinfo или утилиту `makeinfo`, чтобы вставить эти указатели автоматически.)

Не делайте ‘`(dir)`’ ‘Предыдущей’ нодой первой ноды, так как это приводит к непонятному для пользователей поведению программы: если вы находитесь в первой ноде и нажимаете клавишу ⟨DEL⟩, чтобы вернуться назад, вы перейдете к середине какого-то другого вхождения файла ‘`dir`’, не имеющего никакого отношения к тому, что вы читали.

См. Раздел 20.2 [Установка Info-файла], с. 167, для большей информации об установке Info-файла в каталог ‘`info`’.

### 6.3.6 Команда `@top`

Специальная команда `@top` была создана для использования со строкой `@node Top`. Команда `@top` сообщает программе `makeinfo`, что здесь помещена первая нода файла. Она предоставляет сведения, необходимые `makeinfo`, чтобы автоматически вставлять указатели на ноды. Пишите команду `@top` в начале строки сразу после строки `@node Top`. На оставшейся части той же строки напишите заглавие.

В Info команда `@top` выводит на отдельной строке заголовок и строку звездочек под ним.

В TEX и `texinfo-format-buffer`, команда `@top` — это просто синоним команды `@unnumbered`. Обе эти форматирующие программы не требуют наличия команды `@top`

и не делают для нее ничего особеного. Если вы пользуетесь этими программами форматирования, то можете использовать после строки `@node Top` команды `@chapter` или `@unnumbered`. Вы также можете писать `@chapter` или `@unnumbered` при использовании команд обновления Texinfo для создания или обновления указателей и меню.

### 6.3.7 Обзор в ноде Top

Вы можете помочь читателям, написав в первой ноде обзор, после строки `@top` и перед главным меню. В Info этот обзор появится непосредственно перед главным меню. В печатном руководстве этот обзор появится на отдельной странице.

Если вы не хотите, чтобы в печатном руководстве обзор выводился на отдельной странице, вы можете заключить всю первую ноду, включая строки `@node Top` и `@top` или другие команды для разделов, между командами `@ifinfo` и `@end ifinfo`. Это предотвратит появление всего этого текста в печатном выводе. (см. Глава 16 [Условно видимый текст], с. 134). Вы можете повторить краткое описание из первой ноды внутри блока `@iftex ... @end iftex` в начале первой главы, для тех, кто читает печатное руководство. Это сберегает бумагу и может выглядеть красивее.

В обзоре вы должны написать номер версии программы, к которой относится это руководство. Это поможет читателю проследить, какое руководство написано для какой версии программы. Если руководство меняется чаще, чем сама программа, или независимо от нее, вы также должны включить номер редакции для руководства. (Титульный лист тоже должен содержать эти сведения: смотрите Раздел 3.4.1 [@titlepage], с. 35.)

## 6.4 Создание указателей с помощью `makeinfo`

Программа `makeinfo` умеет автоматически определять указатели на ноды для иерархически организованных файлов.

Если вы пользуетесь этой возможностью, вам не нужно писать указатели ‘Next’, ‘Previous’ и ‘Up’ после имени ноды. Однако, вы должны писать команды для задания структуры разделов, такие как `@chapter` или `@section`, на строке, идущей сразу после укороченной строки `@node` (можно лишь написать строку комментария).

Кроме того, после строки `@node` в ноде ‘Top’ вы должны написать строку, начинающуюся командой `@top`, чтобы обозначить первую ноду файла. См. Раздел 5.3 [@top], с. 47.

Наконец, вы должны написать имя каждой ноды (кроме ноды ‘Top’) в меню одним или несколькими иерархическими уровнями выше, чем уровень данной ноды.

Эта способность `makeinfo` вставлять указатели на ноды освобождает вас от необходимости создания и обновления меню и указателей вручную или с помощью команд режима Texinfo. (См. Раздел 2.4 [Обновление нод и меню], с. 19.)

## 6.5 `@anchor`: Определение произвольных назначений для ссылок

*Маркер* — это позиция в вашем документе, помеченная так, что на нее могут указывать перекрестные ссылки, в точности так же, как на ноды. Вы можете создать

маркер, написав команду `@anchor` и задав ему метку в виде обычного аргумента в фигурных скобках. Например:

```
Это @anchor{x-место}место помечено.
...
@xref{x-место■место}.
```

дает:

```
Это место помечено.
...
Смотрите [место], страница 1.
```

Как вы видите, команда `@anchor` сама не дает вывода. В этом примере определен маркер ‘x-место’ непосредственно перед словом ‘место’. Вы можете сослаться на него позже с помощью команды `@xref` или другой команды для перекрестных ссылок, как показано. См. Глава 8 [Перекрестные ссылки], с. 65, подробности о командах для создания перекрестных ссылок.

Лучше всего помещать команды `@anchor` непосредственно перед позицией, на которую вы хотите сослаться; тогда при переходе к маркеру взгляд читателя будет перенесен к нужному фрагменту текста. Вы можете помещать команду `@anchor` на отдельной строке, если это послужит облегчению чтения исходного текста. Пробелы после команды `@anchor` всегда игнорируются.

Имена маркеров и нод не должны конфликтовать. Иногда маркеры и ноды имеют схожее значение; например, в самостоятельной программе Info, команда `goto-node` принимает в качестве аргумента как имя ноды, так и имя маркера. (См. раздел “goto-node” в *GNU Info*.)

# 7 Меню

*Меню* содержат указатели на подчиненные ноды.[1] В Info вы можете использовать меню для перехода к этим нодам. В печатных руководствах меню не нужны и не появляются в них.

По соглашению, меню помещается в конце ноды, так как читатель может не увидеть текст, следующий после меню. Более того, в ноде, содержащей меню, *не должно быть* много текста. Если у вас есть много текста и меню, переместите большую часть текста, кроме нескольких строк, в новую подноду. Иначе читатель, имеющий терминал, неспособный отобразить сразу много строк, может пропустить меню и связанный с ним текст. На практике лучше помещать меню в пределах двадцати строк в начале ноды.

Короткий отрывок текста перед меню может выглядеть некрасиво в печатном руководстве. Чтобы избежать этого, вы можете написать меню рядом с началом его ноды и поместить после него строку `@node`, а затем строку `@heading` между командами `@ifinfo` и `@end ifinfo`. При этом меню, строка `@node` и заголовок появятся только в Info-файле, но не в печатном документе.

Например, два предыдущих абзаца были написаны после меню, строки `@node` и заголовка. В исходном файле это выглядит так:

```
@menu
* Размещение меню::
* Написание меню::
* Части меню::
* Менее беспорядочный пункт меню::
* Пример меню::
* Другие Info-файлы::

@end menu

@node Размещение меню, Написание меню, , Меню
@ifinfo
@heading Меню должны быть в коротких нодах
@end ifinfo
```

Texinfo-файл для данного документа содержит более дюжины примеров применения этой процедуры. Один из них находится в начале этой главы, другой — в начале Глава 8 [Перекрестные ссылки], с. 65.

## 7.1 Написание меню

Меню состоит из команды `@menu`, стоящей на отдельной строке, последующих пунктов меню или строк комментариев к ним и команды `@end menu` на отдельной строке.

[1] Меню могут привести вас к любой ноде, вне зависимости от иерархической структуры, даже к нодам другого Info-файла. Однако, команды обновления режима Texinfo для GNU Emacs могут создавать только меню подчиненных нод. Обычно для обращения к другим нодам применяют перекрестные ссылки.

Описание меню выглядит так:

```
@menu
Более объемные куски текста

* Файлы::                      Все об обращении с файлами.
* Множества::                  Множество буферов; редактирование
                                 нескольких файлов одновременно.
@end menu
```

Каждая строка в меню, начинающаяся с '`* `', является *пунктом меню*. (Обратите внимание на пробел после звездочки.) В меню может появиться и строка, не начинающаяся с '`* `'. Такая строка — это не пункт меню, а комментарий, который появляется в Info-файле. В примере выше строка '`Более объемные куски текста`' является строкой комментария меню; две строки, начинающиеся с '`* `' — пункты меню. Пробелы в меню сохраняются как есть; это позволяет вам форматировать меню по вашему желанию.

## 7.2 Составные части меню

Пункт меню состоит из трех частей, необходимой является только вторая:

1. Название пункта меню (необязательно).
2. Имя ноды (обязательно).
3. Описание пункта (необязательно).

Шаблон пункта меню выглядит следующим образом:

`* `*название-пункта-меню*`: `*имя-ноды*`.  `*описание*

После названия пункта меню должно следовать одно двоеточие, а после имени ноды — символ табуляции, запятая, точка или символ новой строки.

В Info, пользователь выбирает ноду командой `m` (`Info-menu`). Название пункта меню — это то, что пользователь вводит после команды `m`.

В третьей части пункта меню пишется описывающая фраза или предложение. Названия пунктов и имена нод часто бывают короткими; описание же объясняет пользователю, о чем говорится в этой ноде. Хорошее описание дополняет имя ноды, а не просто повторяет его. Описание может находиться на двух или более строках; в этом случае некоторые авторы предпочитают делать отступ во второй строке, тогда как другие предпочитают выравнивать ее по первой (и по остальным) строкам. Здесь выбор за вами.

## 7.3 Менее беспорядочный пункт меню

Если название пункта меню и имя ноды одинаковы, вы можете написать имя сразу после звездочки и пробела в начале строки и поставить после имени два двоеточия.

Например, пишите

`* Имя::`  *описание*

вместо

```
* Имя: Имя.                    описание
```

Вам стоит использовать имя ноды как название пункта меню везде, где это возможно, так как это уменьшит количество ненужного текста в меню.

## 7.4 Пример меню

В Texinfo меню выглядит так:

```
@menu
* название пункта меню: имя ноды.   Короткое описание.
* Имя ноды::                        Эта форма предпочтительна.
@end menu
```

Это дает:

```
* menu:

* название пункта меню: имя ноды.   Короткое описание.
* Имя ноды::                        Эта форма предпочтительна.
```

Пример, как вы можете увидеть это в Texinfo-файле:

```
@menu
Более объемные куски текста

* Файлы::                      Все об обращении с файлами.
* Множества: Буферы.           Множество буферов; редактирование
                                  нескольких файлов одновременно.

@end menu
```

Это дает:

```
* menu:
Более объемные куски текста

* Файлы::                      Все об обращении с файлами.
* Множества: Буферы.           Множество буферов; редактирование
                                  нескольких файлов одновременно.
```

В этом примере меню имеет два пункта. ‘**Файлы**’ — это одновременно и название пункта меню, и имя ноды, на которую ссылается этот пункт. ‘**Множества**’ — это название пункта меню; он ссылается на ноду, называемую ‘**Буферы**’. Строка ‘**Более объемные куски текста**’ является комментарием; она присутствует в меню, но не является его пунктом.

Так как ни для ноды ‘**Файлы**’, ни для ноды ‘**Буферы**’ не указано имени файла, они должны быть именами нод в том же Info-файле (см. Раздел 7.5 [Ссылки на другие Info-файлы], с. 64).

## 7.5 Ссылки на другие Info-файлы

Вы можете создать пункт меню, который позволит пользователю перейти к ноде в другом Info-файле, написав в круглых скобках имя файла непосредственно перед именем ноды. В этом случае вы должны использовать формат пункта меню с тремя частями, что избавит пользователя от необходимости вводить имя файла.

Формат выглядит следующим образом:

```
@menu
* название-первого-пункта:(файл)нода.          описание

* название-второго-пункта:(файл)другая-нода.  описание
@end menu
```

Например, для ссылки непосредственно на ноды ‘`Схема текста`’ и ‘`Перепривязка`’ в *Руководстве по Emacs*, вы написали бы такое меню:

```
@menu
* Схема текста: (emacs)Режим Outline. Основной режим для
                                       редактирования структуры текста.
* Перепривязка: (emacs)Перепривязка.  Как переопределить
                                       значение ключа.
@end menu
```

Если вы не указали имя ноды, а только имя файла, то Info предполагает, что это ссылка на первую (‘Top’) ноду.

Файл ‘`dir`’, в котором содержится главное меню для системы Info, перечисляет в этом меню только имена файлов. Оно переносит вас непосредственно к первой ноде каждого документа Info. (См. Раздел 20.2 [Установка Info-файла], с. 167.)

Например:

```
* Info: (info).          Система просмотра документации.
* Emacs: (emacs).        Расширяемый, самодокументированный
                          текстовый редактор.
```

(Каталог верхнего уровня ‘`dir`’ системы Info — не Texinfo-, а Info-файл, но пункты меню выглядят одинаково в обоих типах файлов.)

Команды обновления режима Texinfo в GNU Emacs работают только с нодами текущего буфера, а значит вы не можете использовать их для создания меню, ссылающихся на другие файлы. Вы должны писать такие меню сами.

# 8 Перекрестные ссылки

*Перекрестные ссылки* используются, чтобы указать читателю на другие части того же или другого Texinfo-файла. В Texinfo перекрестные ссылки могут указывать на ноды и маркеры.

Часто, хоть и не всегда, печатный документ нужно разрабатывать так, чтобы его можно было читать последовательно. Людей утомляет пролистывать вперед и назад, чтобы найти информацию, которая должна быть предоставлена им по мере необходимости.

Однако, в любом документе некоторые сведения могут быть слишком подробными в текущем контексте или мало относящими к нему; используйте для предоставления доступа к такой информации перекрестные ссылки. Кроме того, интерактивная система справок не похожа на роман; немногие читают такие документы последовательно от начала до конца. Напротив, люди ищут то, что им нужно. По этой причине такие произведения должны содержать много перекрестных ссылок, чтобы помочь читателям найти сведения, которые они могли не еще читать.

В печатном руководстве перекрестные ссылки выражаются в виде ссылок на страницы, если только это не ссылка на другое руководство, в этом случае перекрестная ссылка называет это руководство.

В Info, перекрестная ссылка выражается в виде вхождения, по которому вы можете перейти с помощью команды Info ‘`f`’. (См. Info файл ‘`info`’, node ‘`Help-Adv`’.)

Различные команды, работающие с перекрестными ссылками, используют ноды (или маркеры, см. Раздел 6.5 [@anchor], с. 59) для определения позиции, на которую указывает ссылка. Это очевидно в Info, где перекрестная ссылка переносит вас к указанной позиции. TEX также использует ноды для определения назначения перекрестной ссылки, но его действия не столь очевидны. Когда TEX генерирует DVI-файл, он записывает номер страницы каждой ноды и использует эти номера при создании перекрестных ссылок. Таким образом, если вы пишете руководство, предназначенное только для печати, и которое не будет использоваться интерактивно, вы тем не менее должны писать строки `@node` для указания мест, на которые вы будете делать перекрестные ссылки.

## 8.1 Различные команды для перекрестных ссылок

Существует четыре команды для перекрестных ссылок:

`@xref` Используется для начала предложения в печатном руководстве, говорящего ‘Смотрите . . .’, или перекрестной ссылки Info, говорящей ‘`*Note` *имя*`:` *нода*`.’.

`@ref` Используется внутри или, чаще, в конце предложения; то же, что и `@xref` для Info; в печатном руководстве дает только перекрестную ссылку без предшествующего слова ‘Смотрите’.

`@pxref` Используется внутри круглых скобок для создания ссылки, подходящей и для Info-файла, и для печатной книги. В печатном руководстве начинается со слова ‘смотрите’ со строчной буквы. (‘`p`’ означает ‘parenthesis’, то есть ‘круглые скобки’.)

`@inforef` Используется для создания ссылки на Info-файл, для которого нет печатного руководства.

(Команда `@cite` используется для создания ссылок на книги и руководства, для которых нет соответствующего Info-файла, и, следовательно, нет ноды, на которую можно сослаться. См. Раздел 9.1.11 [@cite], с. 83.)

## 8.2 Части перекрестных ссылок

Команда, создающая перекрестную ссылку, требует только один аргумент, имя ноды, на которую указывает ссылка. Но эта команда может содержать до четырех дополнительных аргументов. Используя эти аргументы, вы можете предоставить имя ссылки для Info, описание темы или название раздела для печатного вывода или название другого печатного руководства.

Вот простой пример перекрестной ссылки:

```
@xref{имя ноды}.
```

что дает

```
*Note Имя ноды::.
```

и

Смотрите Раздел *nnn* [имя ноды], стр. *ppp*.

Вот пример полной перекрестной ссылки из пяти частей:

```
@xref{имя ноды, имя перекрестной ссылки, тема, имя-Info-файла,
печатное руководство}, для подробной информации.
```

что дает

```
*Note имя перекрестной ссылки: (имя-Info-файла)имя ноды,
для подробной информации.
```

в Info и

Смотрите раздел “тема” в *печатном руководстве*, для подробной информации.

в печатной книге.

Пять возможных аргументов для перекрестной ссылки включают:

1. Имя ноды или маркера (обязательно). Это позиция, к которой вас переносит перекрестная ссылка. В печатном документе, позиция ноды предоставляет ссылку на страницу только в пределах того же документа.
2. Имя перекрестной ссылки для Info, если оно должно отличаться от имени ноды. Если вы включаете этот аргумент, он становится первой частью перекрестной ссылки. Обычно его опускают.
3. Описание темы или название раздела. Часто это заголовок раздела. Он используется в качестве имени ссылки в печатном руководстве. Если этот аргумент опущен, используется имя ноды.
4. Имя Info-файла, на который указывает ссылка, если он отличен от текущего файла. Вам не нужно включать в имя файла какие-либо суффиксы ‘`.info`’, так как программы чтения Info попытаются добавить его автоматически.
5. Название печатного руководства из другого Texinfo-файла.

Шаблон полной перекрестной ссылки с пять аргументами выглядит так:

`@xref{`*имя-ноды*`,` *имя-перекрестной-ссылки*`,`
*тема-или-раздел*`,` *имя-info-файла*`,`
*название-печатного-руководства*`}.`

Перекрестные ссылки с одним, двумя, четырьмя и пятью аргументами описаны отдельно после `@xref`.

Пишите имя ноды в перекрестной ссылке точно так же, как написано в строке `@node`, включая те же заглавные буквы; иначе программы форматирования могут не найти эту ссылку.

Вы можете написать в абзаце команду, создающую перекрестную ссылку, но помните, как Info и TEX форматируют вывод каждой из них: пишите `@xref` в начале предложения; пишите `@pxref` только внутри круглых скобок и так далее.

## 8.3 Команда `@xref`

Команда `@xref` создает перекрестную ссылку в начале предложения. Команды форматирования для Info преобразуют ее в перекрестную ссылку Info, которую может использовать команда Info '`f`' для перенесения вас непосредственно к другой ноде. Команды набора TEX превращают ее в ссылку на страницу или на другую книгу или руководство.

Чаще всего перекрестная ссылка Info выглядит так:

`*Note` *имя-ноды*`::.`

или так

`*Note` *имя-перекрестной-ссылки*`:` *имя-ноды*`.`

В TEX перекрестная ссылка выглядит так:

Смотрите раздел *номер-раздела* [*имя-ноды*], стр. *страница*.

или так

Смотрите раздел *номер-раздела* [*название-или-тема*], стр. *страница*.

Команда `@xref` не выводит точку или запятую для окончания перекрестной ссылки ни в Info-файле, ни при печати. Вы должны сами поставить точку или запятую; иначе Info не распознает конец перекрестной ссылки. (Команда `@pxref` действует иначе. См. Раздел 8.6 [@pxref], с. 72.)

> **Пожалуйста, обратите внимание:** после закрывающей фигурной скобки `@xref` **обязана** стоять точка или запятая. Это необходимо для завершения перекрестной ссылки. Эта точка или запятая появятся при выводе, как в Info-файле, так и в печатном руководстве.

`@xref` должна ссылаться на ноду Info по имени. Используйте `@node` для определения ноды (см. Раздел 6.3.2 [Написание ноды], с. 56).

После `@xref` идут несколько аргументов в фигурных скобках, разделенные запятыми. Пробельные символы перед запятыми и после них игнорируются.

Перекрестная ссылка требует только имя ноды; но она может содержать до четырех дополнительных аргументов. Каждый из этих вариантов производит перекрестные ссылки, выглядящие несколько по-разному.

**Пожалуйста, обратите внимание:** запятые разделяют аргументы в перекрестной ссылке; избегайте включения их в названия разделов или другие части, чтобы программы форматирования не приняли их за разделители.

### 8.3.1 `@xref` с одним аргументом

Простейшая форма `@xref` принимает один аргумент, имя другой ноды в том же Info-файле. Программы форматирования для Info производят вывод, который программы чтения могут использовать для перехода по ссылке; TEX производит вывод, показывающий вам номер раздела и страницы.

Например,

```
@xref{Тропические бури}.
```

дает

```
*Note Тропические бури::.
```

и

> Смотрите раздел 3.1 [Тропические бури], стр. 24.

(Заметьте, что в этом примере после закрывающей фигурной скобки стоит точка.)

Вы можете написать после перекрестной ссылки дополняющую фразу, как здесь:

```
@xref{Тропические бури}, для подробной информации.
```

что дает

```
*Note Тропические бури::, для подробной информации.
```

и

> Смотрите раздел 3.1 [Тропические бури], стр. 24, для подробной информации.

(Заметьте, что в этом примере после закрывающей фигурной скобки написана запятая, а после дополняющей фразы стоит точка.)

### 8.3.2 `@xref` с двумя аргументами

Если заданы два аргумента, второй используется как имя перекрестной ссылки в Info, тогда как первый так же является именем ноды, на которую указывает эта ссылка.

Шаблон выглядит так:

```
@xref{имя-ноды, имя-перекрестной-ссылки}.
```

Например,

```
@xref{Электрические эффекты, Молния}.
```

дает:

```
*Note Молния: Электрические эффекты.
```

и

> Смотрите раздел 5.2 [Электрические эффекты], стр. 57.

(Заметьте, что в этом примере после закрывающей фигурной скобки стоит точка, и что печатается имя ноды, а не имя перекрестной ссылки.)

Вы можете написать после перекрестной ссылки дополняющую фразу, как здесь:

```
@xref{Электрические эффекты, Молния}, для подробной информации.
```

что дает

```
*Note Молния: Электрические эффекты, для подробной информации.
```

и

> Смотрите раздел 5.2 [Электрические эффекты], стр. 57, для подробной информации.

(Заметьте, что в этом примере после закрывающей фигурной скобки написана запятая, а после дополняющей фразы стоит точка.)

### 8.3.3 `@xref` с тремя аргументами

Третий аргумент замещает имя ноды в выводе TEX. Третий аргумент должен быть названием раздела в печатном выводе или объявлять тему, обсуждаемую в этом разделе. Часто вам стоит писать название с заглавной буквы, чтобы его было легче читать напечатанным. Используйте третий аргумент, когда имя ноды не подходит по синтаксису или по смыслу.

Помните, что нельзя ставить точку внутри названия или темы раздела для перекрестной ссылки или внутри любой другой части команды. Программы форматирования делят перекрестные ссылки на аргументы в соответствии с запятыми; запятая внутри названия или другой части разобьет ее на два аргумента. В ссылке вы должны писать названия вроде “Облака, дымка и туман” без запятых.

Также, не забывайте ставить после закрывающей фигурной скобки `@xref` запятую или точку для завершения перекрестной ссылки. В последующих примерах после завершающей запятой идет дополняющая фраза.

Шаблон выглядит так:

```
@xref{имя-ноды, имя-перекрестной-ссылки,
название-или-тема}.
```

Например,

```
@xref{Электрические эффекты, Молния, Гром и молния},
для деталей.
```

дает

```
*Note Молния: Электрические эффекты, для деталей.
```

и

> Смотрите раздел 5.2 [Гром и молния], стр. 57, для деталей.

Если третий аргумент задан, а второй пуст, то третий служит и для того, и для другого. (Заметьте, как две последовательные запятые помечают пустой второй аргумент.)

```
@xref{Электрические эффекты, , Гром и молния}, для деталей.
```

дает

```
*Note Гром и молния: Электрические эффекты, для деталей.
```

и

> Смотрите раздел 5.2 [Гром и молния], стр. 57, для деталей.

Фактически, часто лучше всего писать перекрестные ссылки лишь с одним аргументом, если имя ноды и название раздела совпадают, или с первым и третьим аргументами, если имя ноды и название раздела различны.

Вот несколько примеров из *The GNU Awk User's Guide*:

```
@xref{Sample Program}.
@xref{Glossary}.
@xref{Case-sensitivity, ,Case-sensitivity in Matching}.
@xref{Close Output, , Closing Output Files and Pipes},
   for more information.
@xref{Regexp, , Regular Expressions as Patterns}.
```

### 8.3.4 `@xref` с четырьмя и пятью аргументами

Четвертый аргумент в перекрестной ссылке задает имя другого Info-файла, отличного от файла, где появляется ссылка, а пятый аргумент задает название для печатного руководства.

Помните, что после закрывающей фигурной скобки `@xref` должна стоять запятая или точка для завершения перекрестной ссылки. В последующих примерах после завершающей запятой идет дополняющая фраза.

Шаблон таков:

```
@xref{имя-ноды, имя-перекрестной-ссылки,
название-или-тема, имя-info-файла,
название-печатного-руководства}.
```

Например,

```
@xref{Электрические эффекты, Молния, Гром и молния, погода,
Введение в метеорологию}, для деталей.
```

дает

```
*Note Молния: (погода)Электрические эффекты, для деталей.
```

Имя Info-файла заключается в круглые скобки и предшествует имени ноды.

В печатном руководстве эта ссылка выглядит так:

> Смотрите раздел "Гром и молния" в книге *Введение в метеорологию*, для деталей.

Название печатного руководства набирается курсивом; в ссылке нет номера страницы, так как TEX не может знать, на какую страницу она указывает, если это ссылка на другое руководство.

Часто вы будете опускать второй аргумент при использовании длинной версии `@xref`. В этом случае в качестве имени ссылки в Info будет использоваться третий аргумент, описание темы.

Шаблон выглядит так:

```
@xref{имя-ноды, , название-или-тема, имя-info-файла,
название-печатного-руководства}, для деталей.
```

что дает

```
*Note название-или-тема: (имя-info-файла)имя-ноды,
для деталей.
```

и

Смотрите раздел *название-или-тема* в книге *название-печатного-руководства*, для деталей.

Например,

```
@xref{Электрические эффекты, , Гром и молния,
погода, Введение в метеорологию}, для деталей.
```

дает

```
*Note Гром и молния: (погода)Электрические эффекты,
для деталей.
```

и

Смотрите раздел “Гром и молния” в книге *Введение в метеорологию*, для деталей.

В редких случаях вы можете захотеть сослаться на другой Info-файл в пределах одного печатного руководства — когда несколько Texinfo-файлов объединяются в одном запуске TEX, но создают раздельные Info-файлы. В этом случае вам нужно указать только четвертый аргумент, но не пятый.

## 8.4 Именование ноды Top

В перекрестной ссылке вы всегда должны называть ноду. Это значит, что для ссылки на все руководство вы должны обозначить ноду ‘Top’, написав ее в качестве первого аргумента команды `@xref`. (Это отличается от способа, которым вы пишите пункты меню; смотрите Раздел 7.5 [Ссылки на другие Info-файлы], с. 64.) В то же время, чтобы предоставить осмысленную тему или название для перекрестной ссылки (вместо слова ‘Top’), вы должны написать подходящий третий аргумент для команды `@xref`.

Таким образом, чтобы создать перекрестную ссылку на *Руководство по GNU Make*, напишите:

```
@xref{Top, , Обзор, make, Руководство по GNU Make}.
```

что дает

```
*Note Обзор: (make)Top.
```

и

Смотрите раздел “Обзор” в книге *Руководство по GNU Make*.

‘`Top`’ в этом примере является именем первой ноды, а ‘`Обзор`’ — названием первого раздела этого руководства.

## 8.5 `@ref`

`@ref` практически та же команда, что и `@xref`, за исключением того, что она не пишет ‘смотрите’ в печатном выводе, а пишет только саму ссылку. Поэтому она удобна в последней части предложения.

Например,

```
Для получения большей информации смотрите @ref{Ураганы}.
```

дает

```
Для получения большей информации смотрите *Note Ураганы::.
```

и

> Для получения большей информации смотрите раздел 8.2 [Ураганы], стр. 123.

Команда `@ref` иногда приводит авторов к изложению своих мыслей в такой форме, которая подходит для печатного руководства, но в формате Info смотрится неудачно. Помните, что ваши читатели будут использовать как печатный формат, так и формат Info.

Например,

```
Морские волны описаны в @ref{Ураганы}.
```

дает

> Морские волны описаны в разделе 6.7 [Ураганы], стр. 72.

в печатном документе, и следующее в Info:

```
Морские волны описаны в *Note Ураганы::.
```

> **Осторожно:** Вы *должны* ставить точку, запятую или правую круглую скобку сразу после команды `@ref` с двумя или более аргументами. Иначе Info не найдет конец перекрестной ссылки и попытка перейти по ссылке завершится неуспехом. В качестве общего правила, вы должны писать точку или запятую после каждой команды `@ref`. Это смотрится лучше всего как в печатном выводе, так и в Info.

## 8.6 @pxref

Команда для перекрестных ссылок в круглых скобках, `@pxref`, почти эквивалентна `@xref`, но вы можете использовать ее *только* внутри круглых скобок, и вы *не* ставите запятую или точку после закрывающей фигурной скобки команды. Эта команда отличается от `@xref` с двух сторон:

1. TEX набирает ссылку для печатного руководства со словом ‘смотрите’ со строчной буквы, а не с заглавной.
2. Команды форматирования Info автоматически завершают ссылку двоеточием или точкой.

Так как один тип форматирования автоматически вставляет завершающую пунктуацию, а другой не вставляет, вы должны использовать `@pxref` *только* внутри круглых скобок, как часть другого предложения. Кроме того, вы не должны сами ставить знаки препинания после ссылки, как вы делаете для `@xref`.

`@pxref` разработана так, чтобы вывод выглядел правильно как в напечатанном виде, так и в Info-файле. В печатном руководстве после перекрестной ссылки в круглых скобках не должна стоять запятая или точка; такая пунктуация неправильна. Но в Info-файле, после перекрестной ссылки должна идти подходящая пунктуация, чтобы Info могла распознать конец ссылки. `@pxref` избавляет вас от необходимости

использования сложных методов поставить ограничитель в одной форме и не ставить в другой.

С одним аргументом, перекрестная ссылка в круглых скобках выглядит так:

```
... бури причиняют наводнения (@pxref{Ураганы}) ...
```

что дает

```
... бури причиняют наводнения (*Note Ураганы::) ...
```

и

... бури причиняют наводнения (смотрите раздел 6.7 [Ураганы], стр. 72) ...

С двумя аргументами, перекрестная ссылка в круглых скобках выглядит так:

```
... (@pxref{имя-ноды, имя-перекрестной-ссылки})...
```

что дает

```
... (*Note имя-перекрестной-ссылки: имя-ноды.) ...
```

и

... (смотрите раздел *nnn* [*имя-ноды*], стр. *ppp*) ...

`@pxref` можно передать до пяти аргументов, в точности, как и `@xref` (см. Раздел 8.3 [@xref], с. 67).

> **Пожалуйста, обратите внимание:** Используйте `@pxref` только для перекрестных ссылок в круглых скобках. Не пытайтесь использовать `@pxref` в качестве дополнения в предложении. Это будет смотреться плохо либо в Info-файле, либо в печатном выводе, либо и там, и там.
>
> Также, перекрестные ссылки в круглых скобках смотрятся лучше всего в конце предложения. Хотя вы можете написать их в середине предложения, такое расположение нарушит течение текста.

## 8.7 `@inforef`

`@inforef` используется для перекрестных ссылок на Info-файлы, для которых нет печатных руководств. Даже в печатном руководстве `@inforef` создает ссылку, направляющую пользователя к Info-файлу.

Эта команда принимает два или три аргумента в следующем порядке:

1. Имя ноды.
2. Имя перекрестной ссылки (необязательно).
3. Имя Info-файла.

Разделяйте аргументы запятыми, как и в `@xref`. Также, вы должны завершить ссылку запятой или точкой после '`}`', как вы делаете для `@xref`.

Шаблон таков:

```
@inforef{имя-ноды, имя-перекрестной-ссылки,
имя-info-файла},
```

Таким образом,

```
@inforef{Эксперт, Продвинутые команды Info, info},
для дальнейшей информации.
```

дает

```
*Note Продвинутые команды Info: (info)Эксперт,
для дальнейшей информации.
```

и

Смотрите Info-файл ‘info’, нода ‘Эксперт’, для дальнейшей информации.

Аналогично,

```
@inforef{Эксперт, , info}, для дальнейшей информации.
```

дает

```
*Note (info)Эксперт::, для дальнейшей информации.
```

и

Смотрите Info-файл ‘info’, нода ‘Эксперт’, для дальнейшей информации.

Обратной к `@inforef` является команда `@cite`, которая используется для ссылки на печатное произведение, не существующее в форме Info. См. Раздел 9.1.11 [@cite], с. 83.

## 8.8 @uref{*url*[, *отображаемый-текст*]}

`@uref` производит ссылку на унифицированный указатель ресурса (url). Она принимает один обязательный аргумент, собственно url, и два необязательных, которые определяют отображаемый текст. При выводе в HTML `@uref` создает ссылку, по которой можно проследовать.

Второй аргумент, если он задан, — это отображаемый текст (по умолчанию это сам url); при выводе в Info или DVI показывается также и url.

С другой стороны, третий аргумент, если он задан, — это тоже отображаемый текст, но тогда url *не* выводится в любом формате. Это полезно, если текст сам по себе дает хорошую ссылку, как в man-странице. Если задан третий аргумент, то второй игнорируется.

Вот простая форма с одним аргументом, где url является и назначением, и текстом ссылки:

```
Официальный ftp-сайт GNU -- это @uref{ftp://ftp.gnu.org/gnu}.
```

дает:

Официальный ftp-сайт GNU — это ftp://ftp.gnu.org/gnu.

Вот пример формы с двумя аргументами:

```
Официальный
@uref{ftp://ftp.gnu.org/pub/gnu, ftp-сайт GNU}
содержит программы и тексты.
```

дает

Официальный ftp-сайт GNU содержит
программы и тексты.

то есть, в Info это выглядит так:

```
Официальный ftp-сайт GNU (ftp://ftp.gnu.org/gnu) содержит
программы и тексты.
```

а в HTML так:

```
Официальный <a href="ftp://ftp.gnu.org/gnu">ftp-сайт GNU</a>
содержит программы и тексты.
```

Пример формы с третьим аргументом:

```
Программа @uref{http://example.org/man.cgi/1/ls■ls(1)} ...
```

дает это:

Программа ls(1) ...

и в HTML:

```
Программа <a href="http://example.org/man.cgi/1/ls">ls(1)</a>  ...
```

Чтобы просто обозначить url, не создавая ссылки, по которой можно перейти, используйте команду `@url` (см. Раздел 9.1.13 [url], с. 83).

Некоторые предпочитают писать url в недвусмысленном формате:

<URL:http://*машина*/*путь*>

Если хотите, вы можете использовать такую форму во входном файле. Мы не считаем необходимым загромождать вывод дополнительными ‘<URL:’ и ‘>’, так как все программы, пытающиеся обнаруживать url в тексте, должны находить их без ‘<URL:’, если они хотят быть полезными.

# 9 Пометка слов и фраз

В Texinfo вы можете пометить слова и фразы множеством способов. Программы форматирования Texinfo используют эту информацию, чтобы определить, как выделить данный текст. Вы можете указать, например, что какое-то слово является определением, метасинтаксической переменной или символом, используемым в программе. Вы также можете несколькими способами обозначить логическое ударение.

## 9.1 Обозначение определений, команд, etc.

В Texinfo есть команды для обозначения того, на какой именно вид объектов ссылается фрагмент текста. Например, метасинтаксические переменные помечаются командой `@var`, а программный код — командой `@code`. Так как фрагменты текста помечаются командами, говорящими, объектами какого сорта они являются, легко изменить способ обработки такого текста программами форматирования Texinfo. (Texinfo — язык *смысловой* разметки, а не *верстки*.)

Например, в печатном руководстве программный код показывается равноширинным шрифтом; `@code` велит TEX набирать данный текст этим шрифтом. Но можно было бы легко поменять способ, которым TEX выделяет код, так, чтобы использовался другой шрифт, и это изменение не затронет способ выделения примеров пользовательского ввода с клавиатуры. Если бы в теле файла использовались непосредственные команды набора, и вы решили сделать такое изменение, вам пришлось бы проверить каждое вхождение, чтобы убедиться, что вы меняете вид кода, а не чего-то другого, что не должно меняться.

Команды выделения могут применяться для извлечения из файла полезной информации, такой как списки функций или имена файлов. Можно, например, написать программу (или макрос клавиатуры) на языке Emacs Lisp, которая вставляла бы вхождение именного указателя после каждого абзаца, содержащего слова или фразы, помеченные определенной командой. Вы могли бы сделать это для того, чтобы составить указатель функций, если вы еще не написали вхождений.

Эти команды служат для множества целей:

`@code{`*пример-кода*`}`
: Обозначает текст, являющийся примером или фрагментом программы.

`@kbd{`*символы-клавиатуры*`}`
: Обозначает текст, вводимый с клавиатуры.

`@key{`*название-клавиши*`}`
: Обозначает обычно используемое название клавиши на клавиатуре.

`@samp{`*текст*`}`
: Обозначает текст, являющийся буквальным примером последовательности символов.

`@var{`*метасинтаксическая-переменная*`}`
: Обозначает метасинтаксическую переменную.

`@env{`*переменная-среды*`}`
: Обозначает имя переменной среды.

`@file{`*имя-файла*`}`
: Обозначает имя файла.

`@command{`*команда*`}`
: Обозначает текст, являющийся именем команды оболочки.

`@option{`*ключ*`}`
: Обозначает ключ командной строки для какой-то программы.

`@dfn{`*термин*`}`
: Обозначает вводимый или определяемый термин.

`@cite{`*ссылка*`}`
: Обозначает название книги.

`@acronym{`*аббревиатура*`}`
: Обозначает аббревиатуру.

`@url{`*унифицированный-указатель-ресурса*`}`
: Обозначает унифицированный указатель ресурса в World Wide Web.

`@email{`*почтовый-адрес*`[, `*отображаемый-текст*`]}`
: Обозначает адрес электронной почты.

### 9.1.1 `@code{`***пример-кода***`}`

Используйте команду `@code`, чтобы обозначить текст, являющийся фрагментом программы и состоящий из полных синтаксических лексем. Заключайте этот текст в фигурные скобки.

Таким образом, вы должны применять `@code` для выражений в программе, названий переменных или функций, используемых в программе, или ключевых слов языка программирования.

Используйте `@code` для имен команд командных языков, похожих на языки программирования, таких как Texinfo. Например, `@code` и `@samp` записаны в исходном Texinfo-файле как '`@code{@@code}`' и '`@code{@@samp}`', соответственно.

Менять регистр слова внутри команды `@code`, когда оно стоит в начале предложения, неправильно. Большинство компьютерных языков регистрозависимы. В Си, например, `Printf` отличается от идентификатора `printf` и, скорее всего, является его неправильной записью. Даже в языках, не учитывающих регистр букв, пользователя смутит различное написание идентификаторов. Выберите одно написание и придерживайтесь его всегда. Если вы не хотите начинать предложение с имени команды, написанного строчными буквами, вам нужно перестроить предложение.

В печатном руководстве `@code` заставляет TEX набирать аргумент в стиле печатной машинки (равноширинным шрифтом). В Info-файле, она велит командам форматирования для Info окружать текст одиночными кавычками.

Например,

```
Используйте @code{ediff-files} для сравнения двух файлов.
```

дает в печатном руководстве следующее:

Используйте `ediff-files` для сравнения двух файлов.

и такое в Info-файле:

```
Используйте ‘ediff-files’ для сравнения двух файлов.
```

Вот несколько случаев, для которых предпочтительно не использовать `@code`:

- Для имен команд оболочки, таких как `ls` (используйте `@command`).
- Для ключей командной строки, таких как ‘`-c`’, когда эти ключи употребляются сами по себе (используйте `@option`).
- Также, полная команда оболочки часто выглядит лучше, если ее набрать с использованием `@samp`, а не `@code`. В таком случае следует выбирать более приятный для глаз вариант.
- Для переменных среды, таких как `TEXINPUTS` (используйте `@env`).
- Для строки короче синтаксической лексемы. Например, если вы пишете о ‘`goto-ch`’, что является только частью имени функции Emacs Lisp `goto-char`, вам следует использовать `@samp`.
- Вообще, когда вы пишете о знаках, используемых в лексеме; не применяйте `@code`, например, когда вы объясняете, какие буквы или печатные знаки могут использоваться в именах функций. (Применяйте `@samp`.) Также вы не должны применять `@code` для обозначения текста, рассматриваемого, как ввод для программы, если этот входной текст не написан на языке, похожем на язык программирования. Например, не нужно использовать `@code` для команд GNU Emacs, вводимых с клавиатуры (используйте вместо этого `@kbd`), хотя вы можете использовать `@code` для имен функций Emacs Lisp, которые вызываются этими командами.

Поскольку `@command`, `@option`, and `@env` были внесены в язык относительно недавно, возможно применение `@code` или `@samp` для имен команд, ключей и переменных среды. Новые команды позволяют выражать разметку более точно, но от использования старых команд нет действительного вреда, и конечно, давно написанные руководства используют их.

### 9.1.2 `@kbd{`*символы-клавиатуры*`}`

Используйте команду `@kbd` для символов ввода, которые печатает пользователь. Например, чтобы сослаться на символы *`M-a`*, напишите

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

а чтобы сослаться на символы *`M-x shell`*, напишите

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

Команда `@kbd` дает в Info такой же результат, как и `@code`, но в печатном руководстве она по умолчанию выводит другим шрифтом (наклонным равноширинным, вместо обычного равноширинного), так что пользователи могут отличить символы, которые предполагается вводить, от тех, что выводит компьютер.

Так как команда `@kbd` применяется в разных руководствах по-разному, вы можете управлять переключением шрифта для нее с помощью команды `@kbdinputstyle`. Эта команда не влияет на вывод Info. Пишите ее в начале строки с одним одним из этих слов в качестве аргумента:

‘`code`’ Всегда использовать для `@kbd` тот же шрифт, что и для `@code`.

‘`example`’ Использовать для `@kbd` другой шрифт только внутри блоков `@example` или в подобных случаях.

‘`distinct`’
: (по умолчанию) Всегда использовать для `@kbd` отличающийся шрифт.

Вы можете вставлять другие `@`-команды внутри фигурных скобок команды `@kbd`. Вот, например, способ оформить команду, которая можно было бы более подробно описать как “нажмите ‘`r`’ и затем клавишу ⟨RET⟩”:

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

Это дает: *r* ⟨RET⟩

Вы также используете команду `@kbd`, если выписываете вводимые буквы по одной; например:

```
Чтобы дать команду @code{logout},
напечатайте символы @kbd{l o g o u t @key{RET}}.
```

Это дает:

Чтобы дать команду `logout`, напечатайте символы *l o g o u t* ⟨RET⟩.

(Этот пример показывает также, что вы можете добавить для ясности пробелы. Если вы действительно хотите упомянуть пробел как один из вводимых символов напишите для него `@key{SPC}`.)

### 9.1.3 `@key{`*название-клавиши*`}`

Используйте команду `@key` для общепринятых названий клавиш на клавиатуре, как например здесь:

```
@key{RET}
```

Вы можете использовать команду `@key` внутри аргумента команды `@kbd`, когда последовательность вводимых символов содержит одну или более клавиш, имеющих название.

Например, чтобы получить *C-x* ⟨ESC⟩, вы написали бы:

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

Вот список рекомендуемых названий клавиш:

| | |
|---|---|
| `SPC` | Пробел |
| `RET` | Возврат каретки |
| `LFD` | Перевод строки (однако, так как большинство современных клавиатур не имеют клавиши "Перевод строки", было бы лучше называть этот символ *C-j*. |
| `TAB` | Табуляция |
| `BS` | Забой |
| `ESC` | Выход |
| `DEL` | Удаление |
| `SHIFT` | Shift |
| `CTRL` | Control |
| `META` | Meta |

Есть тонкости в обращении со словами вроде ‘meta’ или ‘ctrl’, которые являются названиями клавиш-модификаторов. При упоминании символа, в котором используется клавиша-модификатор, такого как *Meta-a*, используйте просто команду `@kbd`, не используйте `@key`; но если вы ссылаетесь на клавишу-модификатор саму по себе, используйте команду `@key`. Например, пишите ‘`@kbd{Meta-a}`’, чтобы получить *Meta-a* и ‘`@key{META}`’, чтобы получить ⟨META⟩.

### 9.1.4 `@samp{`*текст*`}`

Используйте команду `@samp` для обозначения текста, являющегося буквальным примером или ‘образцом’ последовательности символов в файле, строке, образце, etc. Заключайте этот текст в фигурные скобки. Аргумент выводится в одиночных кавычках как в Info-файле, так и в печатном руководстве; кроме того, в печатном руководстве он печатается равноширинным шрифтом.

```
Чтобы найти совпадения с @samp{foo} в конце строки, используйте
регулярное выражение @samp{foo$}.
```

дает

> Чтобы найти совпадения с ‘`foo`’ в конце строки, используйте регулярное выражение ‘`foo$`’.

Всякий раз, когда вы говорите об одиночных символах, используйте `@samp`, если `@kbd` и `@key` не более подходят к ситуации. Также, вы можете использовать `@samp` для полных выражений на Си и для полных команд оболочки — в этом случае `@samp` часто выглядит лучше, чем `@code`. В основном, `@samp` — это вариант для всего, что не покрывается `@code`, `@kbd` или `@key`.

Включайте знаки препинания в фигурные скобки, только если они являются частью задаваемой строки. Пишите знаки препинания вне фигурных скобок, если они являются частью английского текста[1], окружающего строку. В нижеследующем предложении, например, запятые и точки находятся вне фигурных скобок:

```
В английском гласными являются @samp{a}, @samp{e}, @samp{i},
@samp{o}, @samp{u} и иногда @samp{y}.
```

Это дает:

> В английском гласными являются ‘`a`’, ‘`e`’, ‘`i`’, ‘`o`’, ‘`u`’ и иногда ‘`y`’.

### 9.1.5 `@var{`*метасинтаксическая-переменная*`}`

Используйте команду `@var` для обозначения метасинтаксических переменных. *Метасинтаксическая переменная* — это нечто, обозначающее другой фрагмент текста. Например, вы должны использовать метасинтаксическую переменную в документации функции для описания передаваемых этой функции аргументов.

Не используйте `@var` для имен конкретных переменных в языках программирования. Они являются определенными именами из программы, поэтому для них правильным будет использовать `@code` (см. Раздел 9.1.1 [code], с. 77). Например, переменная Emacs Lisp `texinfo-tex-command` — не метасинтаксическая переменная; она правильно форматируется с использованием `@code`.

[1] Или текста на каком-нибудь другом языке. *(Прим. переводчика)*

Также не используйте `@var` для переменных среды; для них необходима команда `@env` (смотрите следующий раздел).

Действие `@var` в Info-файле — изменить регистр аргумента на верхний; в печатном руководстве и HTML — выводить аргумент наклонным шрифтом.

Например,

```
Чтобы удалить файл @var{имя-файла},
напечатайте @samp{rm @var{имя-файла}}.
```

дает

Чтобы удалить файл *имя-файла*, напечатайте '`rm` *имя-файла*'.

(Заметьте, что `@var` может появиться внутри `@code`, `@samp`, `@file`, etc.)

Пишите метасинтаксические переменные полностью строчными буквами, без пробелов, и используйте дефисы, чтобы облегчить чтение. Таким образом, исходный Texinfo-текст для иллюстрации того, как начинать руководство в Texinfo, выглядит так:

```
\input texinfo
@@setfilename @var{имя-info-файла}
@@settitle @var{название-руководства}
```

Это дает:

```
\input texinfo
@setfilename имя-info-файла
@settitle название-руководства
```

В некоторых стилях документации метасинтаксические переменные показывают в угловых скобках, например:

```
..., напечатайте rm <имя-файла>
```

Однако этот стиль не используется в Texinfo. (Вы можете, конечно, если хотите, изменить исходный код '`texinfo.tex`' и команд форматирования для Info, чтобы они выводили в формате `<...>`.)

### 9.1.6 `@env{`*переменная-среды*`}`

Используйте команду `@env` для обозначения переменных среды, используемых во многих операционных системах, включая GNU. Не используйте ее для метасинтаксических переменных; вместо этого используйте `@var` (смотрите предыдущий раздел).

`@env` по действию эквивалентна `@code`. Например:

```
Переменная среды @env{PATH} задает путь поиска для команд.
```

дает

Переменная среды `PATH` задает путь поиска для команд.

### 9.1.7 `@file{`*имя-файла*`}`

Используйте команду `@file` для обозначения текста, являющегося именем файла, буфера или каталога, или именем ноды в Info. Вы также можете использовать эту команду для окончаний имен файлов. Не используйте `@file` для символов в языке программирования; используйте `@code`.

На данный момент `@file` по действию эквивалентна `@samp`. Например,

```
Файлы @file{.el} находятся
в каталоге @file{/usr/local/emacs/lisp}.
```

дает

Файлы ‘.el’ находятся в каталоге ‘/usr/local/emacs/lisp’.

### 9.1.8 @command{*имя-команды*}

Используйте команду @command для обозначения имен команд, таких как ls или cc.

@command по своему действию эквивалентна @code. Например:

```
Команда @command{ls} печатает список содержимого каталога.
```

дает

Команда ls печатает список содержимого каталога.

Вы должны писать название программы шрифтом для обычного текста, если вы употребляете его как новое слово английского языка, например ‘Emacs’ или ‘Bison’.)

При написании полного вызова команды оболочки, как ‘ls -l’, вам следует использовать либо @samp, либо @code по своему усмотрению.

### 9.1.9 @option{*имя-ключа*}

Используйте команду @option для обозначения ключей командной строки, например ‘-l’, ‘-version’ или ‘-output=*имя-файла*’.

@option по действию эквивалентна @samp. Например:

```
Ключ @option{-l} выдает длинный список.
```

дает

Ключ ‘-l’ выдает длинный список.

В таблицах ключи выглядят лучше, если размечать их с помощью @code.

### 9.1.10 @dfn{*термин*}

Используйте команду @dfn для обозначения вводимого или определяемого термина. Применяйте эту команду только в тех отрывках, чья цель — ввести термин, который будет использоваться впоследствии, или с которым читатель должен быть знаком. Просто первое упоминание термина не заслуживает @dfn. Эта команда выводит курсивом в печатном руководстве и в двойных кавычках в Info-файле. Например:

```
Избавление от файла называется @dfn{удалением}.
```

дает

Избавление от файла называется *удалением*.

Как правило, предложение, содержащее определяемый термин, должно быть его определением. Это предложение не обязано явно говорить, что оно является определением, но должно содержать информацию определения — оно должно прояснять значение термина.

### 9.1.11 `@cite{`*ссылка*`}`

Используйте команду `@cite` для названий книг, для которых нет сопутствующего Info-файла. Эта команда выводит курсивом в печатном руководстве и в двойных кавычках в Info-файле.

Если книга написана в Texinfo, лучше использовать команды перекрестных ссылок, чтобы читатель мог легко перейти по такой ссылке в Info. См. Раздел 8.3 [@xref], с. 67.

### 9.1.12 `@acronym{`*аббревиатура*`}`

Используйте команду `@acronym` для аббревиатур, записанных полностью заглавными буквами, таких как NASA. Аббревиатура задается в качестве единственного аргумента в фигурных скобках, например как '`@acronym{NASA}`'. По соображениям стиля или для определенной аббревиатуры вы можете предпочесть использовать точки, как здесь: '`@acronym{Ф.Б.Р.}`'.

В TEX и HTML аргумент печатается несколько уменьшенным шрифтом. В Info или простом текстовом выводе эта команда ничего не меняет.

### 9.1.13 `@url{`*унифицированный-указатель-ресурса*`}`

Используйте команду `@url` для обозначения унифицированных указателей ресурсов в World Wide Web. Она аналогична `@file`, `@var`, etc. и служит только для целей разметки. Она не создает ссылку, по которой вы можете перейти в HTML-выводе (для этого используйте команду `@uref`, см. Раздел 8.8 [@uref], с. 74). Эта команда полезна для url, не существующих на самом деле. Например:

```
Например, это мог бы быть url
@url{http://host.example.org/path}.
```

что дает:

Например, это мог бы быть url
http://host.example.org/path.

### 9.1.14 `@email{`*адрес*`[, `*отображаемый-текст*`]}`

Используйте команду `@email` для обозначения адреса электронной почты. Она принимает один обязательный аргумент, адрес, и один необязательный, отображаемый текст (по умолчанию это сам адрес).

В Info и TEX адрес показывается в угловых скобках после заданного отображаемого текста. В HTML, `@email` создает ссылку '`mailto`', которая обычно вызывает окно составления письма. Например:

```
Присылайте сообщения об ошибках по адресу
@email{bug-texinfo@@gnu.org}.  Присылайте предложения по
@email{bug-texinfo@@gnu.org, тому же адресу}.
```

дает

Присылайте сообщения об ошибках по адресу bug-texinfo@gnu.org.
Присылайте предложения по тому же адресу.

## 9.2 Логическое ударение

Обычно Texinfo изменяет шрифт для пометки слов в тексте в соответствии с категорией, к которой принадлежат эти слова; примером этого служит команда `@code`. Чаще всего это лучший способ пометки слов. Однако, иногда вы можете захотеть выделить текст, не указывая его категорию. В Texinfo для этого есть две команды. Также, в Texinfo есть несколько команд для задания шрифта, которым TEX будет набирать текст. Эти команды не затрагивают Info, и только одна из них, команда `@r`, имеет обычное употребление.

### 9.2.1 `@emph{`*текст*`}` и `@strong{`*текст*`}`

Команды `@emph` и `@strong` используются для обозначения логического ударения; `@strong` сильнее. В печатном выводе `@emph` производит *курсив*, а `@strong` дает **жирный** шрифт.

Например,

```
@quotation
@strong{Осторожно:} @samp{rm * .[^.]*} удаляет @emph{все}
файлы в каталоге.
@end quotation
```

дает в печатном выводе следующее:

> **Осторожно**: ‘`rm * .[^.]*`’ удаляет *все* файлы в каталоге.

и следующее в Info:

```
     *Осторожно*: ‘rm * .[^.]*’ удаляет _все_
     файлы в каталоге.
```

Команда `@strong` используется редко, за исключением случаев, когда нужно пометить нечто, являющееся в действительности типографическим элементом, таким как слово ‘Осторожно’ в предыдущем примере.

В Info-файле, `@emph` окружает текст символами подчеркивания, ‘`_`’, а `@strong` помещает вокруг текста звездочки.

> **Осторожно:** Не используйте `@strong` со словом ‘`Note`’; Info ошибочно примет это за перекрестную ссылку. Используйте вместо этого фразы вроде **Please note** или **Caution**.

### 9.2.2 `@sc{`*текст*`}`: Шрифт маленьких заглавных букв

Используйте команду ‘`@sc`’, чтобы получить текст, набранный в печатном выводе и в HTML шрифтом маленьких заглавных букв, а в Info-файле напечатанный заглавными буквами. Пишите желаемый текст строчными буквами в фигурных скобках, как здесь:

```
@sc{acm} и @sc{ieee} -- технические сообщества.
```

Это дает:

> ACM и IEEE – технические сообщества.

TEX набирает маленькие заглавные буквы так, чтобы буквы на странице не ‘выскакивали на вас’. Это облегчает чтение текста, набранного маленькими заглавными буквами по сравнению с набранным полностью в верхнем регистре, но обычно все-таки

лучше использовать обычный текст смешанного регистра. Команды форматирования Info выводят текст маленьких заглавных букв в верхнем регистре. В HTML текст выводится заглавными буквами, и для его отображения используется меньший шрифт.

Если текст внутри фигурных скобок команды `@sc` в верхнем регистре, TEX набирает его ПОЛНЫМИ ЗАГЛАВНЫМИ БУКВАМИ. Используйте полные заглавные буквы умеренно или не используйте вообще; поскольку бессмысленно помечать командой `@sc` текст, набранный полностью заглавными буквами, `makeinfo` предупреждает о таких случаях.

Вы также можете применять шрифт маленьких заглавных букв для жаргонных слов, таких как ATO (слово из NASA, обозначающее 'abort to orbit').

Есть некие тонкости в использовании шрифта маленьких заглавных букв с жаргонными словами, вроде CDR; это слово применяется в программировании на Лиспе. В таком случае вы должны использовать шрифт маленьких заглавных букв, когда это слово относится ко второму и последующим элементам списка (CDR списка), но должны использовать '`@code`', когда это слово относится к имени функции Лиспа с таким же написанием.

### 9.2.3 Шрифты печати

Texinfo предоставляет четыре команды, изменяющие шрифт в печатном руководстве, но не влияющие на Info-файл. `@i` устанавливает *курсив* (в некоторых версиях TEX используется наклонный шрифт), `@b` устанавливает **жирный** шрифт, `@t` устанавливает `равноширинный` шрифт в стиле печатной машинки, используемый `@code`, а `@r` устанавливает романский шрифт, обычный шрифт, которым печатается весь текст. Все четыре команды действуют на следующий за ними аргумент, заключенный в фигурные скобки.

Частое применение находит только команда `@r`: в примерах программ вы можете использовать команду `@r` для перевода комментариев из равноширинного шрифта в романский. Это смотрится лучше в печатном выводе.

Например,

```
@lisp
(+ 2 2)    ; @r{Складывает два и два.}
@end lisp
```

дает

```
(+ 2 2)    ; Складывает два и два.
```

По возможности избегайте использования трех остальных команд для шрифтов. Если вам понадобилась одна из них, это скорее всего указывает на пробел в языке Texinfo.

# 10 Цитаты и примеры

Цитаты и примеры — это фрагменты текста, состоящие из одного или более полных абзацев, которые выделяются среди остального текста и обрабатываются особо. Обычно для них делается отступ.

В Texinfo вы всегда начинаете цитату или пример, написав `@`-команду в начале отдельной строки, и завершаете написанием команды `@end` также в начале отдельной строки. Скажем, вы начинаете пример, записывая с новой строки команду `@example`, и завершаете написанием команды `@end example` на отдельной строке так же в ее начале.

## 10.1 Команды ограничения блока

Вот команды для цитат и примеров, рассмотренные подробнее в следующих разделах:

`@quotation`
: Обозначает цитируемый текст. Текст заполняется, смещается вправо и печатается по умолчанию романским шрифтом.

`@example`
: Показывает код, команды и подобные вещи. Текст печатается равноширинным шрифтом и смещается вправо, но не заполняется.

`@smallexample`
: Показывает код, команды и подобные вещи. Подобно `@example`, только в TEX эта команда набирает текст более мелким шрифтом для формата `@smallbook`, меньшего, чем формат 8.5 на 11 дюймов.

`@lisp`
: Как `@example`, но специально для иллюстрации кода на Лиспе. Текст печатается равноширинным шрифтом и смещается вправо, но не заполняется.

`@smalllisp`
: То же для `@lisp`, что и `@smallexample` для `@example`.

`@display`
: Показывает иллюстративный текст. Текст смещается вправо и заполняется, шрифт не меняется (то есть по умолчанию используется романский).

`@smalldisplay`
: То же для `@display`, что и `@smallexample` для `@example`.

`@format`
: Как `@display` (текст не заполняется, шрифт не изменяется), но отступ не делается.

`@smallformat`
: То же для `@format`, что и `@smallexample` для `@example`.

Внутри перечисленных конструкций можно использовать команду `@exdent` для отмены смещения строки вправо.

Команды `@flushleft` и `@flushright` служат для выравнивания незаполненного текста по левому или правому полю.

После перечисленных конструкций можно использовать команду `@noindent` для предотвращения смещения вправо последующего текста как нового абзаца.

Для выделения примера или цитаты вы можете использовать с одной из вышеперечисленных конструкций команду `@cartouche`, которая нарисует вокруг них рамку с закругленными углами. См. Раздел 10.11 [Рисование рамок вокруг примеров], с. 91.

## 10.2 `@quotation`

Текст цитаты обрабатывается как обычно за следующими исключениями:

- поля располагаются ближе к центру страницы, таким образом, для всей цитаты делается отступ;
- отступ первой строки абзаца не превышает отступа других строк;
- в печатном выводе пространство между абзацами уменьшается.

> Это пример текста, написанного между командами `@quotation` и `@end quotation`. Команда `@quotation` чаще всего применяется для обозначения того, что данный фрагмент текста взят из другого (настоящего или гипотетического) печатного произведения.

Пишите команду `@quotation` на отдельной строке. Эта строка исчезнет при выводе. Отмечайте конец цитаты строкой, начинающейся командой `@end quotation` и не содержащей ничего больше. Строка `@end quotation` также не выводится. Таким образом, такой текст:

```
@quotation
Это
нечто.
@end quotation
```

дает при выводе

> Это нечто.

## 10.3 `@example`

Команда `@example` применяется для обозначения примера, не являющегося частью текущего текста, например компьютерного вывода и ввода.

```
Это пример текста, записанного между
командой @example
и командой @end example.
В этом тексте сделаны отступы, но не сделано заполнение.

В печатном руководстве этот фрагмент текста набран равноширинным
шрифтом, и существенно наличие дополнительных пробелов и пустых
строк.  В Info-файле аналогичный результат достигается смещением
каждой строки на пять позиций вправо.
```

Пишите команду `@example` в начале отдельной строки. Отмечайте конец примера командой `@end example` также записанной в начале отдельной строки.

Например,

```
@example
mv foo bar
@end example
```

дает

```
mv foo bar
```

Строки, содержащие `@example` и `@end example`, не выводятся. Чтобы вывод смотрелся хорошо, вы должны поместить пустую строку перед командой `@example` и другую после `@end example`. Заметьте, что все пустые строки внутри блока `@example` появятся при выводе.

> **Внимание:** Не используйте символы табуляции внутри текста примера (и вообще в Texinfo-файлах)! TEX воспринимает символы табуляции как один пробел, и это не похоже на табуляцию. Это проблема TEX. (Если необходимо, вы можете использовать *`M-x untabify`* в Emacs для превращения всех символов табуляции в области в пробелы.)

Примеры часто находятся, говоря логически, “в середине” абзаца, и текст, следующий после примера, не должен смещаться вправо. Команда `@noindent` предотвращает отступ во фрагменте текста, как в новом абзаце.

(Для примеров, включенных в предложение, а не отделенных от предшествующего и последующего текста, применяется команда `@code`. См. Раздел 9.1.1 [@code], с. 77.)

## 10.4 `@noindent`

Пример или другое включение может разбить абзац на сегменты. Обычно форматирующие программы делают в тексте, следующем после примера, отступ, как в начале нового абзаца. Однако вы можете предотвратить это, написав перед этим текстом команду `@noindent` в начале отдельной строки.

Например:

```
@example
Это пример
@end example

@noindent
В этой строке нет отступа.  Как вы видите, начало этой строки
смещено до предела влево, как и в следующей строке.  (Весь пример
целиком заключен между @code{@@display} и @code{@@end display}.)
```

дает при выводе

```
Это пример
```

В этой строке нет отступа. Как вы видите, начало этой строки смещено до предела влево, как и в следующей строке. (Весь пример целиком заключен между `@display` и `@end display`.)

При точной настройке числа пустых строк в выводе Info, помните, что строка, содержащая `@noindent`, не производит пустую строку, как и строка `@end example`.

В исходном Texinfo-файле этого руководства перед каждой строкой, говорящей ‘дает’, написана строка с командой `@noindent`.

Не пишите фигурные скобки после команды `@noindent`; они не обязательны, так как `@noindent` используется вне абзаца (см. Приложение I [Синтаксис команд], с. 220).

## 10.5 `@lisp`

Для примеров на Лиспе используется команда `@lisp`. Она является синонимом команды `@example`.

```
Это пример текста, написанного между командами @lisp и
@end lisp.
```

Используйте команду `@lisp` вместо `@example`, чтобы сохранить информацию о природе примера. Это полезно, например, если вы пишите функцию, вычисляющую весь Лисп-код в Texinfo-файле. После этого вы можете использовать Texinfo-файл в качестве библиотеки Лиспа.[1]

Отмечайте конец блока `@lisp` командой `@end lisp`, записанной на отдельной строке.

## 10.6 Команды блоков `@small...`

Кроме обычных команд `@example` и `@lisp` в Texinfo определены "маленькие" команды в стиле примера. Это команды `@smalldisplay`, `@smallexample`, `@smallformat` и `@smalllisp`. Все они разработаны для использования вместе с командой `@smallbook`, заставляющей TEX выводить печатную книгу в формате 7 на 9.25 дюймов, а не в обычном формате 8.5 на 11.

В TEX, команды `@small...` набирают текст для маленького формата `@smallbook` более мелким шрифтом, чем для формата 8.5 на 11 дюймов. Следовательно, многие примеры, содержащие длинные строки, умещаются на более узкой странице формата `@smallbook` без необходимости укорочения. Обе команды набирают текст шрифтом обычного размера, когда вы форматируете для формата размером 8.5 на 11 дюймов; на самом деле, в этой ситуации команды `@small...` определены эквивалентными соответствующим командам без `small`.

В Info команды `@small...` эквивалентны соответствующим командам без `small`.

Помечайте конец блока `@small...` парной командой `@end small...`. Например, пишите для `@smallexample` парную ей `@end smallexample`.

Вот пример тектса, написанный маленьким шрифтом, используемым командами `@smallexample` и `@smalllisp`:

```
...гарантируют, что вы пользуетесь свободой
распространять копии свободного ПО (и получать за это
вознаграждение, если вы того желаете); что вы получаете
исходный код или можете получить его, если захотите;
что вы можете изменять ПО или использовать его части в
новых свободных программах; и что вы знаете, что вы
можете все это делать.
```

Команды `@small...` облегчают подготовку руководств меньшего формата, избавляя вас от необходимости ручного редактирования примеров таким образом, чтобы они умещались на более узких страницах.

[1] Нетрудно расширить Texinfo для аналогичной работы с Си, Фортраном или другими языками.

Обычно печатный документ смотрится лучше, если вы используете в главе только одну из команд, к примеру, или `@example`, или `@smallexample`. Как можно реже используйте оба формата вперемежку.

См. Раздел 19.11 [Печать “маленьких” книг], с. 154, для получения подробной информации о команде `@smallbook`.

## 10.7 `@display` и `@smalldisplay`

Команда `@display` начинает текст, подобный примеру. Она похожа на команду `@example`, только команда `@display` не использует в печатном руководстве равноширинный шрифт. На самом деле, она не задает никакого шрифта, то есть текст выводится тем же шрифтом, каким бы он выводился без команды `@display`.

```
Это пример текста, написанного между командами @display и
@end display. Команда @display делает в тексте
отступы, но не заполняет его.
```

Texinfo также предоставляет команду `@smalldisplay`, которая похожа на `@display` но использует в формате `@smallbook` меньший шрифт. См. Раздел 10.6 [small], с. 89.

## 10.8 `@format` и `@smallformat`

Команда `@format` похожа на команду `@example` за исключением того, что `@format` не сужает поля.

```
Это пример текста,
      написанного между командами
                  @format и @end format.
Как вы видите
       из этого примера, команда
                  @format не заполняет текст.
```

Texinfo также предоставляет команду `@smallformat`, которая похожа на `@format`, но использует в формате `@smallbook` меньший шрифт. См. Раздел 10.6 [small], с. 89.

## 10.9 `@exdent`: Отмена отступа в строке

Команда `@exdent` убирает все отступы, которые могут появиться в строке. Эта команда пишется в начале строки и действует только на текст, следующий за ней на той же строке. Не заключайте текст в фигурные скобки. В печатном руководстве текст на строке `@exdent` выводится романским шрифтом.

`@exdent` обычно применяется в примерах. Таким образом,

```
@example
Эта строка написана после команды @@example.
@exdent В этой строке отменен отступ.
Эта строка написана после строки с отмененным отступом.
@@end example идет на следующей строке.
```

дает

```
     Эта строка написана после команды @example.
```
В этой строке отменен отступ.
```
     Эта строка написана после строки с отмененным отступом.
     @end example идет на следующей строке.
```

На практике команда `@exdent` применяется редко. Обычно отступ отменяется завершением примера и возвратом к обычной ширине страницы.

## 10.10 `@flushleft` и `@flushright`

Команды `@flushleft` и `@flushright` выравнивают концы строк по левому и правому краю страницы, но не заполняют текст. Эти команды пишутся на отдельных строках, без фигурных скобок. Действие команд `@flushleft` и `@flushright` прекращается командами `@end flushleft` и `@end flushright`, написанными на отдельных строках.

Например,

```
     @flushleft
     Этот текст
     прижат влево.
     @end flushleft
```

дает

Этот текст
прижат влево.

`@flushright` создает тип отступов, часто используемый для обратного адреса в письмах. Например,

```
     @flushright
     Вот пример текста, прижатого
     вправо.  Команда @code{@flushright}
     смещает каждую строку к правому краю, но
     оставляет левые концы неровными.
     @end flushright
```

дает

Вот пример текста, прижатого
вправо. Команда `@flushright`
смещает каждую строку к правому краю, но
оставляет левые концы неровными.

## 10.11 Рисование рамок вокруг примеров

Команда `@cartouche` рисует в печатном руководстве рамку с закругленными углами вокруг своего содержимого. Вы можете использовать эту команду для большего выделения примера или цитаты. Например, вы можете написать руководство, в котором какой-нибудь тип примеров окружается рамкой для заострения внимания читателя.

`@cartouche` влияет только на печатное руководство; она не играет роли в других выходных форматах.

Например,

```
@example
@cartouche
% pwd
/usr/local/share/emacs
@end cartouche
@end example
```

заключает пример из двух строк в рамку с закругленными углами в печатном руководстве.

В печатном руководстве этот пример выглядит следующим образом:

```
% pwd
/usr/local/lib/emacs/info
```

# 11 Перечни и таблицы

В Texinfo есть несколько способов создать перечень или таблицу. Перечни могут быть простыми или нумерованными, в таблицах из двух колонок можно выделить элементы первой колонки; поддерживаются также таблицы с многими колонками.

Texinfo автоматически делает отступы в тексте перечней или таблиц и нумерует перечни, если это необходимо. Последнее свойство полезно, если вы изменяете перечень, так как вам не придется перенумеровывать его вручную.

Нумерованные перечни и таблицы начинаются подходящей `@`-командой в начале строки и завершаются соответствующей командой `@end`, написанной на отдельной строке. Команды для таблиц также требуют, чтобы вы написали информацию о стиле форматирования на той же строке, где находится начинающая `@`-команда.

К примеру, нумерованный перечень начинается командой `@enumerate` и завершается командой `@end enumerate`. Начинайте простой перечень командой `@itemize`, за которой следует форматирующая команда, например `@bullet`, и завершайте командой `@end itemize`.

Перед каждым пунктом перечня пишите команду `@item` или `@itemx`.

Вот пример простого перечня различных видов таблиц и перечней:

- Простые перечни с "горошинами" или без них.
- Нумерованные перечни, использующие числа или буквы.
- Таблицы из двух колонок с выделением.

Это нумерованный перечень с теми же пуктами::

1. Простые перечни с "горошинами" или без них.
2. Нумерованные перечни, использующие числа или буквы.
3. Таблицы из двух колонок с выделением.

А это двухколоночная таблица с теми же пунктами и их `@`-командами:

`@itemize` Простые перечни с "горошинами" или без них.

`@enumerate`
: Нумерованные перечни, использующие числа или буквы.

`@table`
`@ftable`
`@vtable` Таблицы из двух колонок, возможно, с внесением в именной указатель.

## 11.1 `@itemize`: создание простых перечней

Команда `@itemize` создает последовательность абзацев с отступами; на левом поле выводится "горошина" или другой значок в начале каждого абзаца, для которого такой значок нужен.

Начинайте простой перечень, записывая команду `@itemize` в начале строки. После этой команды пишите символ или команду Texinfo, создающую значок. Обычно после `@itemize` вы будете писать `@bullet`, но вы также можете использовать `@minus` или любую команду или знак, превращающийся в Info-файле в одиночный знак. Если вы не хотите меток вообще, используйте `@w`. (Когда вы пишите команду-значок, такую как `@bullet` или `@minus`, после команды `@itemize`, вы можете опустить '`{}`'.) Если вы не указываете команду для создания значка, по умолчанию используется `@bullet`.

Пишите текст самих абзацев после `@itemize` вплоть до строки, содержащей команду `@end itemize`.

Перед каждым абзацем, для которого нужен значок на поле, пишите строку, содержащую только команду `@item`. После `@item` можно писать другой текст.

Обычно вы должны помещать перед командой `@item` пустую строку. Это создаст пустую строку в Info-файле. (TEX в любом случае вставит между строк подходящий пропуск.) За исключением случаев, когда пункты очень короткие, пустые строки улучшают внешний вид перечня.

Ниже приведен пример использования команды `@itemize` и порождаемый им вывод. `@bullet` создает в Info символ '`*`', а в TEX круглую точку ("горошину").

```
@itemize @bullet
@item
Какой-то текст.

@item
Какой-то
другой текст.
@end itemize
```

Это дает:

- Какой-то текст.
- Какой-то другой текст.

Простые перечни можно включать в другие перечни. Вот перечень, помеченный дефисами, включенный в перечень, помеченный "горошинами":

```
@itemize @bullet
@item
Первый пункт.

@itemize @minus
@item
Внутренний пункт.

@item
Второй внутренний пункт.
@end itemize

@item
Второй внешний пункт.
@end itemize
```

Это дает:

- Первый пункт.
  - Внутренний пункт.
  - Второй внутренний пункт.
- Второй внешний пункт.

## 11.2 `@enumerate`: создание нумерованных перечней

Команда `@enumerate` похожа на `@itemize` (см. Раздел 11.1 [`@itemize`], с. 93), только помечает пункты последовательными целыми числами или буквами, а не “горошинами”.

Пишите команду `@enumerate` в начале строки. Эта команда не требует аргумента, но ей можно передать в качестве параметра число или букву. Без аргумента, `@enumerate` начинает перечень с номера ‘`1`’. С числовым аргументом, например ‘`3`’, она начинает перечень с этого номера. С аргументом в виде строчной или прописной буквы, например ‘`a`’ или ‘`A`’, она начинает перечень с этой буквы.

Пишете текст нумерованного перечня так же, как и текст простого: помещайте команду `@item` на отдельной строке перед началом каждого абзаца, который должен быть пронумерован. Не пишите другого текста на строке, начинающейся с `@item`.

Между пунктами перечня вам стоит помещать пустые строки. Как правило это облегчает чтение Info-файла.

Вот пример использования `@enumerate` без аргумента:

```
@enumerate
@item
Глубинные причины.

@item
Поверхностные причины.
@end enumerate
```

Это дает:

1. Глубинные причины.
2. Поверхностные причины.

Вот пример с аргументом *3*:

```
@enumerate 3
@item
Предрасполагающие причины.

@item
Подталкивающие причины.

@item
Увековечивающие причины.
@end enumerate
```

Это дает:

3. Предрасполагающие причины.
4. Подталкивающие причины.
5. Увековечивающие причины.

Ниже приведен краткий обзор возможностей. Обзор составлен в виде перечня с использованием `@enumerate` с аргументом `a`.

a. `@enumerate`

   Без аргумента, создает нумерованный перечень начиная с номера 1.

b. `@enumerate` *положительное-целое*

   С числовым (положительным) аргументом, начинает нумерованный перечень с заданного номера. Вы можете использовать это для продолжения перечня, прерванного другим текстом.

c. `@enumerate` *прописная-буква*

   С аргументом в виде прописной буквы, начинает перечень, в котором пункты помечаются буквами начиная с заданной.

d. `@enumerate` *строчная-буква*

   С аргументом в виде строчной буквы, начинает перечень, в котором пункты помечаются буквами начиная с заданной.

Вы также можете делать вложенные нумерованные перечни, подобные схеме текста.

## 11.3 Создание таблиц из двух колонок

Команда `@table` похожа на `@itemize` (см. Раздел 11.1 [@itemize], с. 93), но позволяет задать название или заголовок для каждого пункта. Эта команда используется для создания двухколоночных таблиц и особенно полезна для глоссариев, пояснений и обзоров ключей командной строки.

Пишите команду `@table` в начале строки и на той же строке пишите ее аргумент, который является “обозначающей” командой Texinfo, такой как `@code`, `@samp`, `@var`, или `@kbd` (см. Раздел 9.1 [Обозначения], с. 76). Хотя обычно этим командам нужно задавать аргумент в фигурных скобках, в данном случае вы должны писать имя команды без аргумента, так как аргумент предоставит команда `@item`. Эта команда применяется к тексту, идущему в первой колонке каждого пункта, и определяет стиль выделения этого текста. Например, если вы напишете `@code`, то текст первой колонки будет выделяться командой `@code`.

Вы также можете предпочесть использовать в качестве аргумента для `@table` команду `@asis`. Команда `@asis` ничего не делает; если вы примените ее после `@table`, TEX и команды форматирования для Info будут выводить текст первых колонок не производя выделение (“как есть”).

(Команда `@table` может работать и другими командами, не перечисленными здесь. Однако вы можете использовать только те команды, которые обычно принимают аргументы в фигурных скобках.)

Начинайте каждый пункт таблицы командой `@item` в начале строки. На той же строке где и `@item` пишите текст первой колонки. На следующих строках пишите текст второй колонки. (Для пустого вхождения второй колонки вам ничего не нужно печатать.) Вы можете писать столько строк иллюстративного текста, сколько захотите, даже несколько абзацев. Но только текст из строки с командой `@item` будет помещен в первую колонку, включая все сноски.

Обычно перед строкой `@item` стоит помещать пустую строку. Это создаст пустую строку в Info-файле. За исключением случаев, когда пункты очень короткие, пустые строки смотрятся лучше.

В приведенной ниже таблице текст первой колонки выделяется с помощью команды `@samp`.

```
@table @samp
@item foo
Это текст для
@samp{foo}.


@item bar
Текст для @samp{bar}.
@end table
```

Это дает:

‘`foo`’ Это текст для ‘`foo`’.

‘`bar`’ Текст для ‘`bar`’.

Если вы хотите описать два или более именованных пункта одним блоком текста, используйте команду `@itemx`. (См. Раздел 11.3.2 [@itemx], с. 97.)

### 11.3.1 `@ftable` и `@vtable`

Команды `@ftable` и `@vtable` действуют так же, как и команда `@table`, за исключением того, что `@ftable` автоматически вносит каждый пункт в первой колонке в указатель функций, а `@vtable` — в указатель переменных. Это упрощает задачу создания именных указателей. В именной указатель вносятся только пункты из той строки, где написана команда `@item`, и именно в той форме, в какой они записаны в этой строке. См. Глава 12 [Создание именных указателей], с. 100, для получения подробной информации об именных указателях.

Начинайте таблицу из двух колонок, использующую `@ftable` или `@vtable`, написав в начале строки эту `@`-команду и, на той же строке, команду Texinfo, такую как `@code`, точно так же, как вы написали бы для команды `@table`. Завершайте таблицу командой `@end ftable` или `@end vtable` на отдельной строке.

Пример использования `@table` смотрите в предыдущем разделе.

### 11.3.2 `@itemx`

Используйте в таблице команду `@itemx`, когда для одного пункта у вас есть два или более вхождения в первой колонке, каждое из которых должно появиться на от-

дельной строке. Команда `@itemx` может применяться для всех вхождений, кроме первого; `@itemx` всегда должна идти после команды `@item`. Команда `@itemx` действует точно так же, как и `@item`, но не создает дополнительного вертикального пропуска над текстом первой колонки.

Например,

```
@table @code
@item upcase
@itemx downcase
Две эти команды принимают в качестве аргумента символ
или строку и возвращают соответствующий символ или
строку в верхнем (нижнем) регистре.
@end table
```

Это дает:

`upcase`
`downcase` Две эти команды принимают в качестве аргумента символ или строку и возвращают соответствующий символ или строку в верхнем (нижнем) регистре.

(Обратите также внимание, этот пример иллюстрирует многострочный поясняющий текст в двухколоночной таблице.)

## 11.4 Таблицы из многих колонок

Команда `@multitable` позволяет создавать таблицы с произвольным числом колонок, каждая из которых имеет нужную вам ширину.

Вы определяете ширину колонок в самой строке `@multitable` и пишите каждую строку самой таблицы после команды `@item`, разделяя колонки командой `@tab`. Наконец, `@end multitable` завершает таблицу. Подробности описаны в следующих разделах.

### 11.4.1 Ширина колонок многоколоночных таблиц

Вы можете задать ширину колонок многоколоночной таблицы двумя способами: долей от длины строки или строкой–прототипом. Одновременное применение двух этих методов не поддерживается. В обоих случаях ширины всех колонок задаются в той же строке, где записана команда `@multitable`.

1. Чтобы задать ширины колонок как доли от полной длины строки, напишите `@columnfractions` и десятичные числа (предположительно меньшие единицы) после команды `@multitable`, как здесь:

   ```
   @multitable @columnfractions .33 .33 .33
   ```

   Доли не обязательно должны давать в сумме точно 1.0, как например эти. Это позволяет вам создавать таблицы, которые не требуют полной длины строки. Вы можете писать первой цифрой ноль, если хотите.

2. Для задания строки-прототипа напишите после команды `@multitable` самое длинное вхождение для каждой колонки, заключенное в фигурные скобки. Например:

```
@multitable {текст для первой колонки} {для второй колонки}
```

Тогда первая колонка будет иметь ширину набранной строки ‘текст для первой колонки’, а вторая — ширину ‘для второй колонки’.

Вхождения прототипа не обязательно должны появиться в самой таблице.

Хотя в этом примере мы использовали простой текст, вхождения прототипа могут содержать команды Texinfo; вероятно, особенно полезны будут команды вроде `@code`.

### 11.4.2 Строки многоколоночных таблиц

После команды для `@multitable`, задающей ширину колонок (смотрите предыдущий раздел), вы начинаете каждую строку в теле таблицы командой `@item` и разделяете вхождения колонок командой `@tab`. Разрывы строк не обрабатываются особо в теле таблицы, и вы можете при необходимости разбивать входные строки в исходном файле.

Вот полный пример таблицы из нескольких колонок (текст из *Руководства по GNU Emacs*, см. раздел “Разделение окон” в *Руководство по The GNU Emacs*):

```
@multitable @columnfractions .15 .45 .4
@item Ключ @tab Команда @tab Описание
@item C-x 2
@tab @code{split-window-vertically}
@tab Разделить выбранное окно на два,
находящихся одно над другим.
@item C-x 3
@tab @code{split-window-horizontally}
@tab Разделить выбранное окно на два,
находящихся одно рядом с другим.
@item C-Mouse-2
@tab
@tab На строке режима или полосе прокрутки окна,
разделить это окно.
@end multitable
```

дает:

| Ключ | Команда | Описание |
|---|---|---|
| C-x 2 | `split-window-vertically` | Разделить выбранное окно на два, находящихся одно над другим. |
| C-x 3 | `split-window-horizontally` | Разделить выбранное окно на два, находящихся одно рядом с другим. |
| C-Mouse-2 | | На строке режима или полосе прокрутки окна, разделить это окно. |

# 12 Именные указатели

Используя Texinfo, вы можете создавать именные указатели без необходимости ручной сортировки и упорядочения. Вхождения в именном указателе перечисляются в алфавитном порядке, вместе с информацией о том, где найти рассмотрение каждого вопроса. В печатном руководстве эта информация заключается в номерах страниц. В Info-файле это пункт меню, ведущий в первой ноде, где было внесено это вхождение.

Texinfo предоставляет несколько предопределенных видов именных указателей: указатель функций, указатель переменных, указатель понятий и другие. Вы можете объединять именные указатели или использовать их по другому назначению, отличному от обычного. Если вы захотите, вы можете определить свои собственные именные указатели.

## 12.1 Создание вхождений именного указателя

Когда вы создаете вхождения именного указателя, вам стоит помнить о том, что люди могут искать что-то по разному. Разные люди думают *о разных* словах, когда они что-нибудь ищут. Хороший именной указатель перечисляет вхождения по всем различным словам, которые люди могут использовать для поиска. Например, один читатель может считать очевидным, что двухбуквенные названия именных указателей должны быть перечислены как “Именные указатели, двухбуквенные названия”, так как фраза “Именной указатель” — это общее понятие. Но другой читатель может помнить об особом понятии о двухбуквенных названиях и пытаться найти эту тему как “Двухбуквенные названия именных указателей”. Хороший указатель предоставит оба вхождения и поможет обоим читателям.

Так же, как и верстка, составление именных указателей — это профессиональное искусство, требующее высокой квалификации,тонкости которого непонятны, пока вам самим не доведется им заняться.

См. Раздел 4.1 [Печать именных указателей и меню], с. 43, для получения информации о печати именных указателей в конце книги или создании меню-указателей в Info-файле.

## 12.2 Предопределенные именные указатели

Texinfo предоставляет шесть предопределенных именных указателей:

- *Указатель понятий*, перечисляющий рассматриваемые понятия.
- *Указатель функций*, перечисляющий функции (например точки входа библиотек).
- *Указатель переменных*, перечисляющий переменные (например глобальные переменные библиотек).
- *Указатель клавиш*, перечисляющий команды, вводимые с клавиатуры.
- *Указатель программ*, перечисляющий названия программ.
- *Указатель типов данных*, перечисляющий типы данных (например структуры, определенные в заголовочных файлах).

Не в каждом руководстве нужны все шесть, большинство использует два или три из них. В этом руководстве два именных указателя: указатель понятий и указатель @-команд (который на самом деле является указателем функций, но назван в заголовке указателем команд). Два или более указателя могут быть объединены в один при помощи команды `@synindex` или `@syncodeindex`. См. Раздел 12.4 [Объединение именных указателей], с. 102.

## 12.3 Определение вхождений именных указателей

Данные для составления именного указателя исходят от многих команд, вносящих вхождения, разбросанных по всему исходному Texinfo-файлу. Каждая команда добавляет одно вхождение в некоторый именной указатель; после форматирования указатель будет сообщать номер страницы или имя ноды в качестве ссылки.

Вхождение указателя состоит из вносящей это вхождение команды, расположенной в начале строки, за которой на оставшейся части строки следует само вхождение.

Например, этот раздел начинается следующими пятью вхождениями для указателя понятий:

```
@cindex Добавление вхождений именных указателей
@cindex Вхождения именных указателей
@cindex Именные указатели, вхождения
@cindex Определение вхождений именных указателей
@cindex Создание вхождений именных указателей
```

Каждый из предопределенных именных указателей имеет свою команду для добавления вхождений; для указателя функций это `@findex` и так далее.

Вхождения указателя понятий состоят из текста. Наилучшим способом написать именной указатель будет выбирать вхождения короткие, но ясные. Если это возможно, вхождения будут смотрятся лучше, если писать каждое слово с не заглавной буквы, а так, как оно писалось бы в предложении. (Пишите с заглавной буквы имена собственные и аббревиатуры, которые всегда должны писаться заглавными буквами.) Эти соглашения об использовании заглавных букв мы применяем в большинстве именных указателей в руководствах GNU.

Если вам не удается сделать вхождение кратким и при этом ясным, сделайте его более длинным и ясным, а не коротким и непонятным. Если многие вхождения состоят из нескольких слов, именной указатель может смотреться лучше, если вы будете использовать другое соглашение: писать первое слово каждого вхождения с заглавной буквы. Но не пишите с заглавной буквы регистрозависимые названия, такие как имена функций Лиспа или Си или команды оболочки, это может быть синтаксической ошибкой.

Какого бы соглашение об использовании заглавных букв вы не придерживались, пожалуйста, придерживайтесь его постоянно!

Вхождения в других именных указателях, не в указателе понятий, — это имена символов языков программирования или названия программ; эти названия обычно регистрозависимы, поэтому используйте заглавные и строчные буквы так, как требуется.

По умолчанию вхождения указателя понятий печатаются маленьким романским шрифтом, а вхождения других указателей — маленьким шрифтом, используемым командой `@code`. Вы можете изменить вид части вхождения с помощью обычных команд Texinfo, например `@file` для имен файлов и `@emph` для обозначения логического ударения (см. Глава 9 [Пометка текста], с. 76).

Предопределены шесть команд добавления вхождения к именному указателю:

`@cindex` *понятие*
: Создать вхождение для *понятия* в указателе понятий.

`@findex` *функция*
: Создать вхождение для *функции* в указателе функций.

`@vindex` *переменная*
: Создать вхождение для *переменной* в указателе переменных.

`@kindex` *клавиша*
: Создать вхождение для *клавиши* в указателе клавиш.

`@pindex` *программа*
: Создать вхождение для *программы* в указателе программ.

`@tindex` *тип данных*
: Создать вхождение для *типа данных* в указателе типов данных.

> **Внимание:** Не используйте двоеточия в тексте вхождения. В Info двоеточие отделяет название пункта меню от имени ноды, поэтому двоеточие в самом вхождении введет Info в заблуждение. См. Раздел 7.2 [Составные части меню], с. 62, для получения большей информации о структуре пунктов меню.

На самом деле вы не обязаны использовать предопределенные именные указатели по их обычному назначению. Например, предположим, что вы хотите перечислить некоторые макросы препроцессора Си. Вы можете поместить их в указатель функций вместе с настоящими функциями, просто записывая для них команду `@findex`; затем, при печати “Указателя функций” как ненумерованной главы, вы можете дать ему название “Указатель функций и макросов”, и с точки зрения читателя все будет согласовано. Или вы могли бы поместить макросы вместе с типами данных, записывая для них команду `@tindex` и потом дав этому указателю подходящее название, чтобы читатель вас понял. (См. Раздел 4.1 [Печать именных указателей и меню], с. 43.)

## 12.4 Объединение именных указателей

Иногда вы можете пожелать объединить два разных именных указателя, например указатели функций и понятий, возможно, потому что один из них получится слишком маленьким, если поместить его отдельно, и это будет смотреться довольно неудачно.

Вы можете поместить функции в указатель понятий, записывая для них команду `@cindex` вместо команды `@findex`, и создать согласованное руководство, напечатав указатель понятий под заголовком ‘Указатель понятий и функций’ и не печатая ‘Указатель функций’ вовсе; но это ненадежный метод. Он работает, только если ваш

документ никогда не включается как часть в другой документ, который разрабатывался с отдельным указателем функций; если же вам придется включить его в такой документ, то функции из вашего документа и из другого не будут собраны вместе. Кроме того, чтобы названия функций печатались в указателе понятий правильным шрифтом, вы должны будете заключить каждую из них между фигурными скобками команды `@code`.

### 12.4.1 `@syncodeindex`

Когда вы хотите объединить указатели понятий и функций в один, вы должны перечислять функции с помощью команды `@findex`, а понятия с помощью команды `@cindex`, и использовать команду `@syncodeindex` для перенаправления вхождений указателя функций в указатель понятий.

Команда `@syncodeindex` принимает два аргумента: название именного указателя, подлежащего перенаправлению, и название указателя, в который он перенаправляется. Шаблон выглядит так:

`@syncodeindex` *источник назначение*

Для этой цели именным указателям присвоены двухбуквенные названия:

‘`cp`’ указатель понятий

‘`fn`’ указатель функций

‘`vr`’ указатель переменных

‘`ky`’ указатель клавиш

‘`pg`’ указатель программ

‘`tp`’ указатель типов данных

Пишите команду `@syncodeindex` до или вскоре после строки end-of-header в начале Texinfo-файла. Например, чтобы объединить указатель функций с указателем понятий, напишите следующее:

```
@syncodeindex fn cp
```

Это приведет к тому, что все вхождения, предназначенные для указателя функций, войдут вместо этого в указатель понятий.

Для объединения указателей переменных и функций с указателем понятий напишите следующее:

```
@syncodeindex vr cp
@syncodeindex fn cp
```

Команда `@syncodeindex` помещает все вхождения из указателя-‘источника’ (перенаправляемого именного указателя) напечатанными шрифтом `@code`, заменяя шрифт, используемый по умолчанию для именного указателя, в который перенаправлены вхождения этого. Таким образом, если вы перенаправили имена функций из указателя функций в указатель понятий, все они будут напечатаны шрифтом `@code`, как и ожидалось.

### 12.4.2 `@synindex`

Команда `@synindex` делает почти то же самое, что и команда `@syncodeindex`, но не помещает вхождения из источника со шрифтом `@code`; вместо него она использует романский шрифт. Таким образом, вы должны использовать `@synindex`, когда перенаправляете указатель понятий в указатель функций.

См. Раздел 4.1 [Печать именных указателей и меню], с. 43, для получения информации о печати именного указателя в конце книги или создании меню-указателей в Info-файле.

## 12.5 Определение новых именных указателей

В дополнение к предопределенным именным указателям, вы можете сами определять именные указатели с помощью команд `@defindex` и `@defcodeindex`. Эти команды создают новые `@`-команды для добавления вхождения к именному указателю, которыми вы помечаете вхождения. Команда `@defindex` используются следующим образом:

```
@defindex название
```

Название именного указателя должно быть словом из двух букв, таким как ‘`au`’. Например:

```
@defindex au
```

Здесь определяется новый именной указатель, называемый ‘`au`’. В то же время создается новая команда добавления к указателю, `@auindex`, которую вы можете использовать для создания вхождений в именной указатель. Новая команда добавления к указателю используется точно так же, как и предопределенные команды.

Например, вот заголовок раздела, за которым следует вхождение указателя понятий и два вхождения указателя ‘`au`’.

```
@section Познавательная семантика
@cindex Схемы кинестетических отображений
@auindex Джонсон, Марк
@auindex Лакоф, Джордж
```

(Очевидно, в данном примере ‘`au`’ служит сокращением слова “author” (автор).) Texinfo конструирует новую команду добавления к именному указателю, приписывая к названию указателя строку ‘`index`’; таким образом, определение именного указателя ‘`au`’ приводит к автоматическому созданию команды `@auindex`.

Для печати именного указателя используйте команду `@printindex`, как и для предопределенных именных указателей. Например:

```
@node Author Index, Subject Index, , Top
@unnumbered Авторы

@printindex au
```

Команда `@defcodeindex` похожа на `@defindex`, но печатает вхождения шрифтом `@code`, а не романским. То есть она создает аналог команды `@findex`, а не `@cindex`

Вы должны определять новые именные указатели внутри или сразу после строки Texinfo-файла end-of-header, до любой команды `@synindex` или `@syncodeindex` (см. Раздел 3.2 [Заголовок файла], с. 29).

# 13 Специальные вставки

Texinfo предоставляет несколько команд для вставки символов, имеющих в Texinfo особое значение, таких как фигурные скобки, и для других графических элементов, не совпадающих с простыми литерами, которые вы можете напечатать.

К ним относятся:

- Фигурные скобки и ‘@’.
- Пробелы внутри и вокруг предложения.
- Акценты.
- Многоточие и “горошины”.
- Лого TEX и знак охраны авторских прав.
- Знак фунта стерлингов.
- Знак минус.
- Математические выражения.
- Значки вычисления, макрорасширения, ошибки, etc.
- Сноски.
- Изображения.

## 13.1 Вставка @ и фигурных скобок

‘@’ и фигурные скобки являются в Texinfo специальными символами. Чтобы вставить эти символы так, чтобы они появились в тексте, вы должны помещать перед ними ‘@’, чтобы уберечь Texinfo от неправильной их интерпретации.

Не помещайте после этих команд фигурные скобки, они не нужны.

### 13.1.1 Вставка ‘@’ с помощью @@

@@ обозначает один символ ‘@’ при выводе как на печать, так и в Info.

Не пишите после команды @@ фигурные скобки.

### 13.1.2 Вставка ‘{’ и ‘}’с помощью @{ и @}

@{ обозначает один символ ‘{’ при выводе как на печать, так и в Info.

@} обозначает один символ ‘}’ при выводе как на печать, так и в Info.

Не пишите после команд @{ и @} фигурные скобки.

## 13.2 Вставка пробелов

Последующие разделы описывают команды, которые управляют разного рода пробелами внутри или вокруг предложений.

### 13.2.1 Незавершение предложения

В зависимости от того, стоит ли точка или восклицательный или вопросительный знак в середине или в конце предложения, в печатном руководстве после них вставляется меньший или больший пробел. Поскольку не всегда можно определить, когда точка завершает предложение, а когда используется для сокращения, в некоторых обстоятельствах требуются специальные команды. Обычно Texinfo может догадаться, как обрабатывать точки, так что вам необязательно использовать специальные команды; просто вводите точки так же, как если бы вы использовали печатную машинку, то есть ставьте **два пробела** после точки, вопросительного или восклицательного знака, завершающего предложение.

Используйте команду `@:` после точки, вопросительного или восклицательного знака или двоеточия, после которых не должен стоять дополнительный пробел. Например, используйте `@:` после точек, завершающих сокращения и не стоящих в конце предложения.

Например,

```
S.o.p.@: состоит из трех частей ...
S.o.p. состоит из трех частей ...
```

дает следующее. Если вы внимательно приглядитесь к напечатанному результату, то увидите во второй строке несколько больший пробел после ‘`s.o.p.`’.

S.o.p. состоит из трех частей ...
S.o.p. состоит из трех частей ...

(Кстати, ‘`s.o.p.`’ — это сокращение от “Standard Operating Procedure” (стандартная процедура действий).)

`@:` не влияет на вывод Info. Не пишите фигурные скобки после `@:`.

### 13.2.2 Завершение предложения

Используйте `@.` вместо точки, `@!` вместо восклицательного знака и `@?` вместо вопросительного знака в конце предложения, заканчивающегося отдельной заглавной буквой. Иначе TEX сочтет эту букву сокращением и не вставит правильный пробел для конца предложения. Вот пример:

```
Дать это M.I.B. и M.E.W@.  Также, дать это R.J.C@.
Дать это M.I.B. и M.E.W.  Также, дать это R.J.C.
```

дает нижеследующее. Если вы внимательно приглядитесь к напечатанному результату, то увидите в первой строке несколько больший пробел после ‘`W`’.

Дать это M.I.B. и M.E.W. Также, дать это R.J.C.
Дать это M.I.B. и M.E.W. Также, дать это R.J.C.

При выводе в Info-файл `@.` эквивалентна просто ‘`.`’; аналогично для `@!` и `@?`.

Командам Texinfo `@:` и `@.` даны именно такие имена, чтобы они хорошо работали с командами Emacs для перемещения по предложениям. (см. раздел “Предложения” в *Руководство по GNU Emacs*).

Не пишите фигурные скобки после любой из этих команд.

### 13.2.3 Несколько пробелов

Обычно TEX сворачивает повторяющиеся пробельные символы (пробел, табуляция и новая строка) в один пробел. Вывод для Info, с другой стороны, сохраняет пропуски такими, как вы их напечатали, только заменяет перевод строки на пробел; именно поэтому важно помещать два пробела в конце предложений в документах Texinfo.

Иногда вы можете захотеть действительно вставить несколько пробелов подряд для примера (что делает ваша программа с несколькими введенными пробелами) или просто для улучшения внешнего вида заголовков перечней. Texinfo поддерживает три команды: `@SPACE`, `@TAB` и `@NL`, каждая из которых вставляет на выходе один пробел. (Здесь `@SPACE` представляет символ '`@`', за которым идет пробел, то есть '`@ `', а `TAB` и `NL` представляют символы табуляции и символ конца строки, то есть когда '`@`' стоит последним на строке.)

Например,

```
Пример@ @ @ @
с пробелами.
```

дает

```
Пример    с пробелами.
```

Другие возможные применения `@SPACE` реализует команда `@multitable` (см. Раздел 11.4 [Многоколоночные таблицы], с. 98).

Не пишите фигурные скобки после любой из этих команд.

### 13.2.4 `@dmn{`*размер*`}`: Форматирование размеров

Иногда вам может понадобиться написать '`12 pt`' или '`8.5 in`' с маленьким пробелом или без пробела между числом и сокращенным названием единицы измерения. Для этого вы можете использовать команду `@dmn`. Встретив эту команду, TEX вставляет точно необходимый для правильного набора пропуск; команды форматирования Info не вставляют пробела совсем, так как он не требуется в Info-файле.

Чтобы применить команду `@dmn`, напишите число и сразу после него, без промежуточного пробела, `@dmn`, а затем единицу измерения в фигурных скобках. Например,

```
Формат A4 имеет ширину 8.27@dmn{in}.
```

дает

Формат A4 имеет ширину 8.27 in.

Не все придерживаются этого стиля. Некоторые предпочитают писать в Texinfo-файле '`8.27 in.@:`' или '`8.27 дюймов`' вместо '`8.27@dmn{in}`'. Однако, в таких случаях форматирующие программы могут разорвать строку между числом и единицей измерения, поэтому используйте `@w` (см. Раздел 14.3 [w], с. 118). Также, если вы пишите точку после сокращения в середине предложения, вы должны написать после нее '`@:`', чтобы TEX не вставил дополнительный пробел, как показано здесь. См. Раздел 13.2.1 [Незавершение предложения], с. 106.

## 13.3 Вставка акцентов

Ниже приведена таблица команд, предоставляемых Texinfo для вставки плавающих акцентов. Команды с небуквенными именами не требуют фигурных скобок вокруг аргумента (которым считается следующий символ). (Исключение: `@,` *требует* фигурные скобки вокруг аргумента.) Это сделано для того, чтобы исходный текст было читать как можно удобнее, так как символы с акцентами очень часты в некоторых языках.

| Команда | Вывод | Описание |
|---|---|---|
| `@"o` | ö | умляут |
| `@'o` | ó | ударение |
| `@,{c}` | ç | седиль |
| `@=o` | ō | макрон или надчеркивание |
| `@^o` | ô | сиркомфлекс |
| `@`o` | ò | грав акцент |
| `@~o` | õ | тильда |
| `@dotaccent{o}` | ȯ | точка сверху |
| `@H{o}` | ő | длинный венгерский умляут |
| `@ringaccent{o}` | o | кружок сверху |
| `@tieaccent{oo}` | o͡o | лига |
| `@u{o}` | ŏ | акцент краткости |
| `@ubaraccent{o}` | o̲ | подчеркивание |
| `@udotaccent{o}` | ọ | точка снизу |
| `@v{o}` | ǒ | хачек или галочка |

Эта таблица перечисляет команды Texinfo для вставки символов, часто используемых в языках, отличных от английского.

| | | |
|---|---|---|
| `@exclamdown{}` | ¡ | перевернутый ! |
| `@questiondown{}` | ¿ | перевернутый ? |
| `@aa{},@AA{}` | a,A | a,A с кружком |
| `@ae{},@AE{}` | æ,fl | лигатуры ae,AE |
| `@dotless{i}` | ı | i без точки |
| `@dotless{j}` | ȷ | j без точки |
| `@l{},@L{}` | ł,Ł | перечеркнутая L,l |
| `@o{},@O{}` | fi,ffl | перечеркнутая O,o |
| `@oe{},@OE{}` | ff,ffi | лигатуры oe,OE |
| `@ss{}` | ı | эс-цет или острое S |

## 13.4 Вставка многоточий и “горошин”

*Многоточие* (последовательные точки) набирается не как строка точек, поэтому для многоточия в Texinfo используется специальная команда. Команда `@bullet` также является специальной. После каждой из этих команд пишется пара фигурных скобок, ‘`{}`’, без пробела между именем команды и скобками. (Вы должны писать с этими командами фигурные скобки, потому что вы можете использовать их сразу после другого текста; при отсутствии скобок форматирующие программы могут запутаться. См. Приложение I [Синтаксис @-команд], с. 220, для дальнейшей информации.)

### 13.4.1 `@dots{}` (...) и `@enddots{}` (....)

Используйте команду `@dots{}` для создания многоточия, то есть трех точек подряд, с подходящими пробелами между ними, как здесь: '. . .'. Не пишите во входном файле просто три точки; это сработает при выводе в Info-файл, но в печатном руководстве создаст неправильные пробелы между точками.

Аналогично, команда `@enddots{}` создает многоточие, завершающее предложение (четыре точки)....

Вот многоточие: . . . Вот три точки подряд: ...

В печатном выводе три точки подряд расположены ближе друг к другу, чем точки в многоточии.

### 13.4.2 `@bullet{}` (•)

Используйте команду `@bullet{}` для создания большой круглой точки ("горошины") или наиболее похожего символа. В Info используется звездочка.

Вот "горошина": •

Если вы используете `@bullet` в `@itemize`, вам не нужно печатать фигурные скобки, так как их предоставит `@itemize`. (См. Раздел 11.1 [@itemize], с. 93.)

## 13.5 Вставка TEX и ©

Лого 'TEX' набирается особым образом, и для него требуется `@`-команда. Символ охраны авторских прав, '©', также является специальным. После каждой из этих команд пишется пара фигурных скобок, '`{}`', без пробела между именем команды и скобками.

### 13.5.1 `@TeX{}` (TEX)

Используйте команду `@TeX{}` для вывода 'TEX'. В печатном руководстве это особое лого, отличающееся от трех простых букв. В Info оно выглядит как '`TeX`'. Команда `@TeX{}` уникальна среди других команд Texinfo тем, что в ней есть заглавные буквы '`T`' и '`X`'.

### 13.5.2 `@copyright{}` (©)

Используйте команду `@copyright{}` для вывода '©'. В печатном руководстве это '`c`' в круге, а в Info это '`(C)`'.

## 13.6 `@pounds{}` ($): Фунты стерлингов

Используйте команду `@pounds{}` для вывода '$'. В печатном руководстве это символ, обозначающий фунт стерлингов. В Info это '`#`'. Символы других денежных единиц, к сожалению, недоступны.

## 13.7 `@minus{}` (−): Вставка знака минус

Используйте команду `@minus{}` для вывода знака минус. В равноширинном шрифте это один дефис, но в пропорциональном шрифте это обычно символ с длиной знака минус — немного длиннее дефиса, но короче em-тире:

‘−’ это знак минус, созданный командой ‘`@minus{}`’,

‘-’ это дефис, созданный с помощью символа ‘`-`’,

‘—’ это em-тире для текста.

В равноширинном шрифте, используемом Info, `@minus{}` аналогичен дефису.

Вы не должны применять `@minus{}` внутри блока `@code` или `@example`, потому что в равноширинном шрифте, который они используют, не делается различий по ширине.

Когда вы используете `@minus` для задания значка, начинающего каждый пункт перечня, вам не нужно печатать фигурные скобки (см. Раздел 11.1 [`@itemize`], с. 93).

## 13.8 `@math`: Вставка математических выражений

Вы можете записать короткие математические выражения с помощью команды `@math`. Пишите выражение в фигурных скобках, как здесь:

```
@math{(a + b)(a + b) = a^2 + 2ab + b^2}
```

Это дает в TEX следующее:

$$(a + b)(a + b) = a^2 + 2ab + b^2$$

и следующее в Info:

```
(a + b)(a + b) = a^2 + 2ab + b^2
```

Таким образом, команда `@math` не влияет на вывод Info.

Для сложных математических выражений вы также можете использовать TEX напрямую (см. Раздел 16.3 [Команды прямого форматирования], с. 135). Когда вы используете TEX непосредственно, не забывайте писать математические выражения между одинарными или двойными символами ‘`$`’ (знаками доллара), как необходимо.

## 13.9 Графические знаки для примеров

В Texinfo, код часто иллюстрируется примерами, ограниченными командами `@example` и `@end example` или `@lisp` и `@end lisp`. В таких примерах вы можете обозначить результат вычисления или раскрытия с помощью ‘⇒’ или ‘↦’. Аналогично, есть команды для вставки знаков, обозначающих печатаемый вывод, сообщения об ошибках, эквивалентность выражений и позицию точки.

Команды вставки графических знаков необязательно использовать внутри примера, но чаще всего бывает так. После каждой команды для вставки графического знака следует пара фигурных скобок.

⇒ `@result{}` указывает на результат выражения.

↦ `@expansion{}` показывает результат раскрытия макроса.

| | |
|---|---|
| ⊣ | `@print{}` обозначает печатаемый вывод. |
| error | `@error{}` обозначает, что последующий текст — это сообщение об ошибке. |
| ≡ | `@equiv{}` обозначает точную эквивалентность двух форм. |
| ⋆ | `@point{}` показывает позицию точки. |

### 13.9.1 `@result{}` (⇒): Обозначение вычисления

Используйте команду `@result{}` для обозначения результата вычисления выражения.

Команда `@result{}` отображается как '`=>`' в Info и как '⇒' в печатном выводе.

Таким образом, следующий пример:

```
(cdr '(1 2 3))
     ⇒ (2 3)
```

можно прочитать как "`(cdr '(1 2 3))` при вычислении дает `(2 3)`".

### 13.9.2 `@expansion{}` (↦): Обозначение раскрытия

Когда выражение является вызовом макроса, оно раскрывается в новое выражение. Вы можете обозначить результат раскрытия с помощью команды `@expansion{}`.

Команда `@expansion{}` отображается в Info как '`==>`', а в печатном выводе как '↦'.

Например, следующее

```
@lisp
(third '(a b c))
     @expansion{} (car (cdr (cdr '(a b c))))
     @result{} c
@end lisp
```

дает

```
(third '(a b c))
     ↦ (car (cdr (cdr '(a b c))))
     ⇒ c
```

что можно прочитать так:

> `(third '(a b c))` раскрывается в `(car (cdr (cdr '(a b c))))`; результат вычисления этого выражения равен `c`.

Часто, как в этом случае, пример смотрится лучше, если для команд `@expansion{}` и `@result{}` сделан отступ в пять пробелов.

### 13.9.3 `@print{}` (⊣): Обозначение печатаемого вывода

Иногда выражение печатает что-то во время исполнения. Вы можете обозначить печатаемый вывод с помощью команды `@print{}`.

Команда `@print{}` отображается в Info как '`-|`', а в печатном выводе как '⊣'.

В следующем примере печатаемый текст обозначен с помощью '⊣', а результат выражения идет на следующей строке.

```
(progn (print 'foo) (print 'bar))
     ⊣ foo
     ⊣ bar
     ⇒ bar
```

В исходном Texinfo-файле этот пример записан следующим образом:

```
@lisp
(progn (print 'foo) (print 'bar))
     @print{} foo
     @print{} bar
     @result{} bar
@end lisp
```

### 13.9.4 `@error{}` (error): Обозначение сообщения об ошибке

Фрагмент кода может вызвать сообщение об ошибке, когда вы его вычисляете. Вы можете обозначить сообщение об ошибке с помощью команды `@error{}`.

Команда `@error{}` отображается в Info как '`error->`', а в печатном выводе как 'error '.

Таким образом,

```
@lisp
(+ 23 'x)
@error{} Wrong type argument: integer-or-marker-p, x
@end lisp
```

дает

```
(+ 23 'x)
error  Wrong type argument: integer-or-marker-p, x
```

Это обозначает, что когда вы вычисляете выражение, печатается сообщение об ошибке:

```
Wrong type argument: integer-or-marker-p, x
```

сама строка 'error ' не является частью сообщения об ошибке.

### 13.9.5 `@equiv{}` (≡): Обозначение эквивалентности

Иногда два выражения выдают одинаковый результат. Вы можете обозначить точную эквивалентность двух форм с помощью команды `@equiv{}`.

Команда `@equiv{}` отображается в Info как '`==`', а в печатном выводе как '≡'.

Таким образом,

```
@lisp
(make-sparse-keymap) @equiv{} (list 'keymap)
@end lisp
```

дает

```
(make-sparse-keymap) ≡ (list 'keymap)
```

Это обозначает, что вычисление `(make-sparse-keymap)` дает тот же результат, что и вычисление `(list 'keymap)`.

### 13.9.6 `@point{}` (⋆): Обозначение точки в буфере

Иногда вам может понадобиться показать пример текста в буфере Emacs. В таких примерах действует соглашение включать все содержимое рассматриваемого буфера между двух строк из дефисов, содержащих имя этого буфера.

Вы можете использовать команду '`@point{}`', чтобы показать позицию точки в тексте буфера. (Символ точки, конечно, не является частью текста буфера; он обозначает место *между* двумя символами, где находится точка.)

Команда `@point{}` отображается в Info как '`-!-`', а в печатном выводе как '⋆'.

Следующий пример показывает содержимое буфера '`foo`' до и после вычисления Лисп-команды, вставляющей слово **измененное**.

```
------- Buffer: foo -------
Это ⋆содержимое foo.
------- Buffer: foo -------

(insert "измененное ")
     ⇒ nil
------- Buffer: foo -------
Это измененное ⋆содержимое foo.
------- Buffer: foo -------
```

В исходном Texinfo-файле этот пример был записан так:

```
@example
------- Buffer: foo -------
Это @point{}содержимое foo.
------- Buffer: foo -------

(insert "измененное ")
     @result{} nil
------- Buffer: foo -------
Это измененное @point{}содержимое foo.
------- Buffer: foo -------
@end example
```

## 13.10 Сноски

*Сноски* нужны для справок, дополняющих или поясняющих основной текст.[1]

### 13.10.1 Команды создания сносок

В Texinfo сноски создаются командой `@footnote`. Сразу за командой следует открывающая фигурная скобка, потом текст сноски, и за ним закрывающая фигурная скобка. Сноски могут быть произвольной длины (при необходимости сноски разбиваются по страницам), но обычно они бывают короткими. Вот шаблон:

[1] Сноски могут дополнять или расширять основной текст, но читатель должен иметь возможность понять текст, не читая сносок. Подробное обсуждение применения сносок вы можете прочитать в книге *The Chicago Manual of Style*, опубликованной издательством *University of Chicago Press*.

```
обычный текст@footnote{текст сноски}
```

Как здесь показано, команда `@footnote` должна следовать сразу после поясняемого текста; иначе знак сноски может появиться на новой строке.

Например, после этой фразы вставлен образец сноски[2]; в исходном Texinfo-файле это выглядит так:

```
...образец сноски@footnote{Это пример
сноски.}; в исходном Texinfo-файле...
```

В печатном руководстве или книге знак сноски — это маленькое приподнятое число; текст сноски появляется внизу страницы под горизонтальной чертой.

В Info знак сноски — это номер сноски в круглых скобках, например: '`(1)`'. После знака сноски следует перекрестная ссылка к тексту сноски.

В HTML файле, знаки сноски изображаются как маленькие приподнятые числа, служащие гиперссылками к тексту сноски.

Кстати, сноски в аргументе команды `@item` внутри таблиц должны быть на той же строке, что и `@item` (как правило). См. Раздел 11.3 [Двухколоночные таблицы], с. 96.

### 13.10.2 Стили сносок

В Info есть два стиля отображения сносок, которые определяют, где появится текст сноски:

- Все сноски в стиле 'В конце' для одной ноды располагаются с конце этой ноды. Они отделяются от остальной части ноды строкой дефисов со словом '**Сноски**' внутри. Каждая сноска начинается со знака сноски '(*n*)'.

  Вот пример одной сноски в стиле 'в конце ноды':

  ```
  ------ Сноски ------

  (1)  Это пример сноски.
  ```

- Все сноски в стиле 'Отдельно' для одной ноды автоматически помещаются в отдельную ноду. При использовании этого стиля за каждым знаком сноски '(*n*)' в теле ноды следует "ссылка на сноску". Ссылка на сноску, на самом деле, — это перекрестная ссылка, которую вы используете для перехода к ноде сносок.

  Имя ноды для сносок составляется из имени ноды, в которой определены сноски и строки '`-Footnotes`'. (А значит, нода со сносками к ноде '`Footnotes`' называется '`Footnotes-Footnotes`'!) Нода со сносками имеет ссылку 'Up' (Вверх), указывающую на ее родительскую ноду.

  После форматирования для Info в отдельном стиле, сноска из первой главы этого руководства будет выглядеть так:

  ```
  File: texinfo.info  Node: Overview-Footnotes, Up: Overview

  (1) Обратите внимание, первый слог в■Texinfo■ произносится
  как ■тек■, а не ■текс■. ...
  ```

[2] Это пример сноски.

Texinfo-файл может быть отформатирован в Info-файл с любым стилем сносок.

Используйте команду `@footnotestyle` для задания стиля сносок в Info-файле. Пишите эту команду с начала отдельной строки с последующим аргументом, '`end`' или '`separate`', для помещения сносок соответственно в конце ноды или в отдельной ноде.

Например,

```
@footnotestyle end
```

или

```
@footnotestyle separate
```

Пишите команду `@footnotestyle` перед строкой end-of-header или немного после, в начале Texinfo-файла. (Если вы поместите команду `@footnotestyle` между строк start-of-header и end-of-header, команды форматирования области будут использовать заданный стиль сносок.)

Если вы не задали стиль сносок, команды форматирования будут использовать стиль по умолчанию. На данный момент команды `texinfo-format-buffer` и `texinfo-format-region` используют стиль 'separate', а `makeinfo` — стиль 'end'.

В этой главе есть две сноски.

## 13.11 Вставка рисунков

Вы можете вставить рисунок из внешнего файла с помощью команды `@image`:

`@image{`*имя-файла*`,` [*ширина*]`,` [*высота*]`}`

Аргумент *имя-файла* является обязательным, и имя не должно содержать расширения, потому что разные обработчики поддерживают разные форматы:

- TEX читает файл '*имя-файла*`.eps`' (формат Encapsulated PostScript).
- PDFTEX считывает '*имя-файла*`.pdf`' (Portable Document Format, формат фирмы Adobe).
- `makeinfo` использует '*имя-файла*`.txt`' как есть для вывода в формате Info (более или менее, как если бы это был блок `@example`).
- При выводе в формате HTML, `makeinfo` пытается найти файл '*имя-файла*`.png`'; если такой не существует, она пробует '*имя-файла*`.jpg`'. Если нет и этого файла, она ссобщит об ошибке. (Мы не можем поддерживать формат GIF из-за патентов.)

Необязательные аргументы *ширина* и *высота* задают размер для масштабирования изображения (они игнорируются при выводе в формате Info). Если они оба не заданы, рисунок представляется в естественном размере (заданном в самом файле); если задан только один, второй пропорционально масштабируется; и если заданы оба, то рассматриваются оба, что возможно исказит первоначальный рисунок, изменив отношение его сторон.

Аргументы *ширина* и *высота* могут быть заданы с помощью правильных размеров TEX, а именно:

| | |
|---|---|
| pt | пункт (72.27pt = 1in) |
| pc | пика (1pc = 12pt) |
| bp | большой пункт (72bp = 1in) |

| | |
|---|---|
| in | дюйм |
| cm | сантиметр (2.54cm = 1in) |
| mm | миллиметр (10mm = 1cm) |
| dd | дидот-пункт (1157dd = 1238pt) |
| cc | цицеро (1cc = 12dd) |
| sp | масштабный пункт (65536sp = 1pt) |

Например, следующая команда масштабирует файл ‘`ridt.eps`’ до одного дюйма по вертикали и пропорционально по горизонтали:

```
@image{ridt■1in}
```

Чтобы `@image` работала с TEX, нужно установить файл ‘`epsf.tex`’ в таком месте, где TEX может его найти. (Стандартное место — ‘*texmf*`/tex/generic/dvips/epsf.tex`’, где *texmf* — это корень дерева каталогов TEX.) Этот файл включен в дистрибутив Texinfo и доступен на `ftp://tug.org/tex/epsf.tex`.

`@image` можно использовать как внутри строки, так и для выделенных рисунков. Поэтому, если вы намереваетесь сделать рисунок выделенным, убедитесь, что вы оставили перед этой командой пустую строку, иначе вывод наползет на предшествующий текст.

# 14 Задание и предотвращение разрывов.

Обычно Texinfo-файл обрабатывается и TEX, и одной из команд форматирования для Info. Иногда в том или в другом выходном файле разрывы строк, абзацев или страниц оказываются в 'неправильных' местах. Вы должны убедиться, что текст выглядит хорошо как в печатном руководстве, так и в Info-файле.

Например, в печатном руководстве страница может быть разбита в середине примера, что неудобно; чтобы предотвратить это, вы можете сохранить целостность фрагмента текста, используя команды группирования, которые удерживают текст от разделения между двумя страницами. Напротив, вы можете захотеть принудительно разбить страницу там, где обычно это не случается. К счастью, подобные проблемы возникают нечасто. Если они все же возникли, используйте команды создания или предотвращения разрывов.

Команды разрыва создают или разрешают разрывы строк и абзацев:

`@*` Принудительный разрыв строки.

`@sp` *n* Пропустить *n* пустых строк.

`@-` Обозначить возможный перенос.

`@hyphenation{`*пе-ре-но-сы слова*`}`
Обозначить возможные места переносов в слове.

Команды предотвращения разрыва строки сохраняют целостность текста в пределах одной строки:

`@w{`*текст*`}` Предотвращает разбиения между строк и переносы в *тексте*.

Команды разбиения на страницы относятся только к печатному руководству, так как в Info-файлах нет страниц.

`@page` Начать новую страницу в печатном руководстве.

`@group` Отметить фрагмент текста, который должен появляться целиком на одной странице печатного руководства.

`@need` *тысячные дюйма*
Начать новую страницу, если на текущей недостаточно места.

## 14.1 @*: Разрыв строки

Команда `@*` принудительно разрывает строку в печатном руководстве и в документе Info.

Например,

```
Эта строка @* разбита @*в двух местах.
```

дает

```
Эта строка
 разбита
в двух местах.
```

(Заметьте, что пробел после первой команды `@*` перенесен на следующую строку.)

Команда `@*` часто используется на странице с информацией об авторских правах:

```
Это редакция 2.0 документации Texinfo,@*
для Texinfo ...
```

в этом примере команда `@*` запрещает TEX некрасиво растягивать строку по всей странице.

**Обратите внимание:** Не пишите фигурные скобки после команды `@*`, они не нужны.

Не пишите в конце абзаца, содержащего команды `@*` команду `@refill`; это приведет к перезаполнению абзаца после разбиения строк и потере результата разбиения.

## 14.2 `@-` и `@hyphenation`: Переносы

Хотя алгоритм переносов в TEX обычно работает довольно хорошо, время от времени он пропускает полезные точки переноса. (Или, намного реже, делает неправильный перенос.) Тогда, для документов с необычным словарем или при окончательной доводке печатного издания, вы можете захотеть помочь TEX. В Texinfo для этого предусмотрены две команды:

`@-` Обозначить возможный перенос, то есть место, где TEX может (но не обязан) сделать перенос. Это особенно полезно, если появляется переполненный бокс, из-за того что TEX не произвел допустимый перенос (см. Раздел 19.10 [Переполненные боксы], с. 154). TEX не будет вставлять других точек переноса в слово, содержащее `@-`.

`@hyphenation{`*пе-ре-но-сы слова*`}`
: Сообщить TEX, как можно сделать переносы в слове. Как показано, вы помещаете '`-`' в каждой точке переноса. Например:

  ```
  @hyphenation{ма-ну-скрипт ма-ну-скрип-ты}
  ```

  TEX использует заданные таким образом точки переноса, только если слова точно совпадают, поэтому вы должны перечислить все необходимые варианты.

В документах Info переносы не используются, а значит эти команды не имеют значения для них.

## 14.3 `@w{`*текст*`}`: Предотвращение разрывов строк

Конструкция `@w{`*текст*`}` выводит *текст* и запрещает разрывы строк внутри *текста*.

Вы можете использовать команду `@w`, чтобы запретить TEX автоматически переносить длинное название, имя или фразу, которые оказались близко к концу строки. Например:

```
Вы можете скопировать программы GNU с
@w{@samp{ftp.gnu.org}}.
```

дает

Вы можете скопировать программы GNU с '`ftp.gnu.org`'.

Вы также можете использовать `@w` для получения неразрывного пробела:

```
Ни одна из форматирующих программ не станет разбивать строку на
этом@w{ }пробеле.
```

## 14.4 `@sp` *n*: Вставка пустых строк

Строка, начинающаяся с `@sp` *n* и не содержащая ничего больше, создает *n* пустых строк в печатном руководстве и в Info-файле. `@sp` также принудительно обрывает абзац. Например,

```
@sp 2
```

создает две пустых строки.

Команда `@sp` чаще всего применяется в титульном листе.

## 14.5 `@page`: Переход на новую страницу

Строка, содержащая только `@page`, начинает новую страницу в печатном руководстве. Эта команда не играет роли для Info-файлов, так как они не разбиваются на страницы. Команда `@page` часто применяется на титульном листе в Texinfo-файле для начала страницы с информацией об авторских правах.

## 14.6 `@group`: Предотвращение разрывов страниц

Команда `@group` (на отдельной строке) применяется внутри конструкции `@example` или подобной для начала неделимой вертикальной группы, появляющейся в печатном документе целиком на одной странице. Группа завершается строкой, содержащей только `@end group`. Две эти команды сами по себе не производят никакого вывода и не играют роли в Info-файле.

Хотя команда `@group` в принципе должна работать во многих контекстах, ее текущая реализация работает надежно только внутри блоков `@example` и его вариантов и внутри `@display`, `@format`, `@flushleft` и `@flushright`. См. Глава 10 [Цитаты и примеры], с. 86. (Все эти команды имеют общую черту — каждая входная строка дает одну строку на выходе.) В других контекстах команда `@group` может привести к ненормальному распределению вертикального свободного места.

Это требование означает, что вы должны писать:

```
@example
@group
...
@end group
@end example
```

где команды `@group` и `@end group` расположены между командами `@example` и `@end example`.

Команда `@group` чаще всего применяется, если нужно сохранить фрагмент текста целиком на одной странице. В этом руководстве по Texinfo более 100 примеров содержат текст, заключенный между `@group` и `@end group`.

Если вы забудете закончить группу, то при запуске TEX вы можете получить странные и непонятные сообщения об ошибках. Это происходит, потому что TEX продолжает пытаться поместить остаток Texinfo-файла на одну страницу и не выдает ошибок, пока не обработает довольно значительный кусок текста. Хорошим практическим правилом будет поискать пропущенную команду `@end group`, если вы получаете от TEX непонятные сообщения.

## 14.7 `@need` *mils*: Предотвращение разрывов страниц

Строка, содержащая только `@need` *n*, начинает в печатном руководстве новую страницу, если на текущей странице осталось места меньше, чем *n* мил (тысячных долей дюйма). Не пишите фигурные скобки вокруг аргумента *n*. Команда `@need` *n* не имеет значения в Info-файлах, так как они не разбиваются на страницы.

Перед этим абзацем написана команда `@need`, сообщающая TEX, что нужно начать новую страницу, если на текущей осталось меньше 800 мил (восьми десятых дюйма). Это выглядит так:

```
@need 800
Перед этим абзацем написана ...
```

Команда `@need` полезна для предотвращения появления изолированных строк (то есть строк, оказавшихся единственными в конце страницы).

# 15 Команды для определений

Команда `@deffn` и другие *команды для определений* позволяют вам описывать функции, переменные, макросы, команды, пользовательские параметры, специальные формы и другие объекты в едином формате.

В Info-файле, определение вызывает появление в начале первой строки определения категории объекта — 'Функция', 'Переменная' или другой, за которой идет имя объекта и его аргументы. В печатном руководстве, эта команда велит TEX печатать имя объекта и его аргументы с левого края, а название его категории — с правого. В обоих выходных форматах для тела определения делается отступ. Также, имя объекта вносится в соответствующий именной указатель: `@deffn` вносит имя в указатель функций, `@defvr` в указатель переменных и так далее.

Руководство не должно и не может содержать более одного определения для заданного имени. Приложение, содержащее обзор, должно использовать `@table`, а не команды определений.

## 15.1 Шаблон определения

Команда `@deffn` используется для определений объектов, похожих на функции. Чтобы записать определение с помощью `@deffn`, напишите команду `@deffn` в начале строки и продолжите строку категорией объекта, именем объекта и его аргументами (если они есть). Потом напишите тело определения на последующих строках. (Вы можете вставлять в тело примеры.) И наконец, завершайте определение командой `@end deffn`, записанной на отдельной строке. (Другие команды для определений придерживаются того же формата.)

Шаблон определения выглядит так:

```
@deffn категория имя аргументы...
тело-определения
@end deffn
```

Например,

```
@deffn Команда forward-word счетчик
Эта команда перемещает точку вперед на столько слов, сколько задано
аргументом @var{счетчик} (или назад, если @var{счетчик}
отрицателен). ...
@end deffn
```

дает

**forward-word** *счетчик* Команда

Эта команда перемещает точку вперед на столько слов, сколько задано аргументом *счетчик* (или назад, если *счетчик* отрицателен). . . .

Пишите название категории с заглавной буквы. Если название категории содержит пробелы, как, скажем, 'Интерактивная команда', заключайте его в фигурные скобки. Например:

```
@deffn {Интерактивная команда} isearch-forward
...
@end deffn
```

Иначе второе слово будет ошибочно принято за имя объекта.

Некоторые команды для определений имеют более общий смысл, чем другие. Команда `@deffn`, например, — это общая команда для функций и похожих объектов: для объектов, которые могут принимать аргументы. Когда вы используете эту команду, вы задаете категорию, к которой принадлежит данный объект. Команда `@deffn` предлагает три предопределенных специализированных варианта, `@defun`, `@defmac` и `@defspec`, которые задают для вас категории “Функция”, “Макро” и “Специальная форма”, соответственно. (В Лиспе специальной формой называется объект, во многом похожий на функцию.) Команда `@defvr` также сопровождается несколькими предопределенными специализированными вариантами для описания конкретных типов переменных.

Шаблон для специализированного определения, такого как `@defun`, похож на шаблон общего определения, за исключением того, что вам не нужно задавать категорию:

```
@defun имя аргументы...
тело-определения
@end defun
```

Таким образом,

```
@defun buffer-end флаг
Эта функция возвращает @code{(point-min)}, если @var{флаг}
меньше 1, и @code{(point-max)} в противном случае.
...
@end defun
```

дает

**buffer-end** *флаг* Function

Эта функция возвращает `(point-min)`, если *флаг* меньше 1, и `(point-max)` в противном случае. ...

См. Раздел 15.6 [Пример определения функции], с. 132, для получения более подробного примера определения функции, включающего использование `@example` внутри определения.

Другие специализированные команды работают подобно `@defun`.

## 15.2 Необязательные и повторяющиеся аргументы

Некоторые объекты принимают необязательные или повторяющиеся аргументы, которые можно обозначить характерным знаком, использующим квадратные скобки и многоточие. Например, специальная форма часто разбивает свой список аргументов на отдельные аргументы более сложным образом, чем прямолинейная функция.

Аргумент, заключенный в квадратные скобки является необязательным. Таким образом, фраза ‘[*аргумент*]’ означает, что *аргумент* необязателен. Аргумент, после которого стоит многоточие, является необязательным и может быть повторен несколько раз. Таким образом, ‘*аргумент*...’ обозначает ноль или более аргументов. Круглые

скобки используются когда несколько аргументов сгруппированы в дополнительные уровни структуры списка в языке Лисп.

Вот, к примеру, строка `@defspec` для воображаемой специальной формы:

**foobar** (*var* [*from to* [*inc*]]) *body*... Special Form

В этом примере аргументы *from* и *to* являются необязательными, но должны либо оба присутствовать, либо оба отсутствовать. Если они присутствуют, также может быть задан *inc*. Эти аргументы сгруппированы с аргументом *var* в один список, чтобы отличить их от *body*, который включает все остальные элементы формы.

В исходном Texinfo-файле эта строка `@defspec` записана следующим образом (за исключением того, что он не была разбита на две строки, как в этом примере).

```
@defspec foobar (@var{var} [@var{from} @var{to}
     [@var{inc}]]) @var{body}@dots{}
```

Эта функция вносится в указатель команд и переменных под именем '`foobar`'.

## 15.3 Две или более "первых" строк

Чтобы создать две или более 'первые' строки, напишите после первой строки `@deffn` строку, начинающуюся с `@deffnx`. Команда `@deffnx` работает в точности, как и `@deffn`, но не создает дополнительный вертикальный пропуск между ней и предыдущей строкой.

Например,

```
@deffn {Интерактивная команда} isearch-forward
@deffnx {Интерактивная команда} isearch-backward
Две эти команды аналогичны, за исключением ...
@end deffn
```

дает

**isearch-forward** Интерактивная команда
**isearch-backward** Интерактивная команда

Две эти команды аналогичны, за исключением ...

Каждая команда для определений имеет форму с 'x': `@defunx`, `@defvrx`, etc.

Форма с 'x' действует так же, как и `@itemx`; смотрите Раздел 11.3.2 [@itemx], с. 97.

## 15.4 Команды для определений

Texinfo предоставляет более дюжины команд для определений; все они описаны в этом разделе.

Команды для определений автоматически вносят имя объекта в соответствующий именной указатель: например, `@deffn`, `@defun` и `@defmac` вносят имена функций в указатель функций; `@defvr` и `@defvar` вносят имена переменных в указатель переменных.

Хотя последующие примеры иллюстрируют преимущественно Лисп, эти команды могут быть использованы и для других языков программирования.

### 15.4.1 Функции и похожие объекты

В этом разделе рассматриваются команды для описания функций и похожих объектов:

`@deffn` *категория имя аргументы...*

Команда `@deffn` — это общая команда для функций, интерактивных команд и похожих объектов, которые могут принимать аргументы. Вы должны выбрать термин для описания категории, к которой принадлежит определяемый объект; например, если объект является функцией, то можно использовать категорию “Функция”. Команда `@deffn` записывается в начале строки, а после нее на той же строке пишутся категория определяемого объекта, имя объекта и его аргументы, если таковые имеются. Завершайте определение с помощью `@end deffn` на отдельной строке.

Например, вот определение:

```
@deffn Команда forward-char nchars
Переместить точку назад на @var{nchars} символов.
@end deffn
```

Этот пример показывает довольно сжатое определение “команды”, называемой `forward-char`, с одним аргументом, *nchars*.

`@deffn` печатает имена аргументов, таких как *nchars*, курсивом или заглавными буквами, как при использовании `@var`, потому что мы думаем об этих именах, как о метасинтаксических переменных — они обозначают значения действительных аргументов. Внутри текста описания, чтобы сослаться на значение аргумента, пишите имя аргумента с явной командой `@var`. В примере выше мы использовали таким образом ‘`@var{nchars}`’.

Шаблон для `@deffn` выглядит так:

```
@deffn категория имя аргументы...
тело-определения
@end deffn
```

`@defun` *имя аргументы...*

Команда `@defun` — это команда для определений функций. `@defun` эквивалентна команде ‘`@deffn Функция ...`’.

Например,

```
@defun set символ новое-значение
Изменить значение символа @var{символ} на
@var{новое-значение}.
@end defun
```

показывает довольно сжатое определение функции `set` с аргументами *символ* и *новое-значение*. Имена аргументов в строке `@defun` автоматически выводятся курсивом или заглавными буквами, как если бы они были заключены в `@var`. Завершайте определение с помощью `@end defun` на отдельной строке.

Шаблон таков:

```
@defun имя-функции аргументы...
тело-определения
@end defun
```

`@defun` создает вхождение в указателе функций.

`@defmac` *имя аргументы...*

Команда `@defmac` — это команда для определений макросов. `@defmac` эквивалентна '`@deffn Макро ...`' и работает подобно `@defun`.

`@defspec` *имя аргументы...*

Команда `@defspec` — это команда для определений специальных форм. (В Лиспе специальной формой называется объект, во многом похожий на функцию, см. раздел "Special Forms" в *GNU Emacs Lisp Reference Manual*.) `@defspec` эквивалентна '`@deffn {Специальная форма} ...`' и работает подобно `@defun`.

### 15.4.2 Переменные и похожие объекты

Ниже перечислены команды для определений переменных и похожих объектов:

`@defvr` *категория имя*

Команда `@defvr` — это общая команда для определений объектов, схожих с переменными — объектов, в которых записано значение. Вы должны выбрать термин для описания категории определяемого объекта; это может быть, например, "Переменная", если объект является переменной. Пишите команду `@defvr` в начале строки и, за ней на той же строке, категорию объекта и его имя.

Пишите название категории с заглавной буквы, как заголовок. Если название категории содержит пробелы, как например "Пользовательский параметр", заключайте его в фигурные скобки. Иначе второе слово будет ошибочно принято за имя объекта. Например,

```
@defvr {Пользовательский параметр} fill-column
Эта локальная переменная задает максимальную
ширину заполненных строк.
...
@end defvr
```

Завершайте определение с помощью `@end defvr` на отдельной строке.

Шаблон таков:

```
@defvr категория имя
тело-определения
@end defvr
```

`@defvr` создает для *имени* вхождение в указателе переменных.

`@defvar` *имя*

Команда `@defvar` — это команда для определений переменных. `@defvar` эквивалентна команде '`@defvr Переменная ...`'.

Например:

```
@defvar kill-ring
...
@end defvar
```

Шаблон таков:

```
@defvar имя
тело-определения
@end defvar
```

`@defvar` создает для *имени* вхождение в указателе переменных.

`@defopt` *имя*

Команда `@defopt` — это команда для определений *пользовательских параметров*, то есть переменных, предназначенных для изменения пользователем по его вкусу; их много в Emacs (см. раздел “Переменные” в *Руководство по GNU Emacs*). `@defopt` эквивалентна ‘`@defvr {Пользовательский параметр} ...`’ и работает подобно `@defvar`.

### 15.4.3 Функции в языках с контролем типов

Команда `@deftypefn` и ее варианты предназначены для описания функций в языках, в которых вы должны объявлять типы переменных и функций, таких как Си и Си++.

`@deftypefn` *категория тип-данных имя аргументы...*

Команда `@deftypefn` — это общая команда для определений функций и похожих объектов, которые могут принимать аргументы и имеют тип. Команда `@deftypefn` пишется в начале строки, и за ней на той же строке следуют название категории описываемого объекта, тип возвращаемого им значения, имя этого объекта и его аргументы, если таковые имеются.

Например,

```
@deftypefn {Библиотечная функция} int foobar
   (int @var{foo}, float @var{bar})
...
@end deftypefn
```

(где текст перед “...”, показанный выше как две строки, на самом деле был бы в настоящем Texinfo-файле на одной строке) дает в Info следующее:

```
- Библиотечная функция: int foobar (int FOO, float BAR)
...
```

В печатном руководстве, это дает:

`int` **foobar** (`int` *foo*, `float` *bar*) Библиотечная функция

...

Это означает, что `foobar` является “библиотечной функцией”, которая возвращает `int`, а ее аргументы — это *foo* (`int`) и *bar* (`float`).

Имена аргументов, которые вы пишете в `@deftypefn`, не передаются неявно `@var` — так как действительные имена аргументов в `@deftypefn` обычно разбросаны между именами типов данных и ключевыми словами, и Texinfo не может найти их без вашей помощи. Вместо этого вы должны явно писать `@var` вокруг имен аргументов. В примере выше, имена аргументов — это ‘`foo`’ и ‘`bar`’.

Вот шаблон для `@deftypefn`:

```
@deftypefn категория тип-данных имя аргументы ...
тело-определения
@end deftypefn
```

Заметьте, что если *категория* или *тип-данных* занимает более одного слова, они должны быть заключены в фигурные скобки, чтобы стать одним аргументом.

Если вы описываете процедуру в языке, в котором есть пакеты, таком как Ада, вам стоит использовать `@deftypefn` способом, несколько противоречащим описанным в предыдущих абзацах соглашениям.

Например:

```
@deftypefn stacks private push
        (@var{s}:in out stack;
        @var{n}:in integer)
...
@end deftypefn
```

(Аргументы `@deftypefn` показаны разбитыми на три строки, но в действительном Texinfo-файле должны быть одной строкой.) В этом случае процедура классифицируется как принадлежащая к пакету `stacks`, а не как ‘процедура’, и ее тип описывается как `private`. (Имя этой процедуры `push`, а ее аргументы — это *s* и *n*.)

`@deftypefn` создает для *имени* вхождение в указателе функций.

`@deftypefun` *тип-данных имя аргументы...*

Команда `@deftypefun` — это специализированная команда для определений функций в языках с контролем типов. Эта команда эквивалентна ‘`@deftypefn Функция ...`’.

Таким образом,

```
@deftypefun int foobar (int @var{foo}, float @var{bar})
...
@end deftypefun
```

дает в Info следующее:

```
- Функция: int foobar (int FOO, float BAR)
...
```

и следующее печатном руководстве:

`int` **foobar** (`int` *foo*, `float` *bar*) Function

...

Вот шаблон:

```
@deftypefun тип имя аргументы...
тело-определения
@end deftypefun
```

`@deftypefun` создает для *имени* вхождение в указателе функций.

### 15.4.4 Переменные в языках с контролем типов

Переменные в языках с контролем типов обрабатываются методом, похожим метод для функций в таких языках. См. Раздел 15.4.3 [Типизированные функции], с. 126. Общая команда для определений `@deftypevr` соответствует `@deftypefn`, а специализированная команда `@deftypevar` соответствует `@deftypefun`.

`@deftypevr` *категория тип-данных имя*

Команда `@deftypevr` — это общая команда для определений объектов, похожих на переменные в языках с контролем типов — объектов, в которых записано значение. Вы должны выбрать термин для описания категории определяемого объекта; это может быть, например, “Переменная”, если объект является переменной.

Команда `@deftypevr` записывается в начале строки, и за ней на той же строке следуют название категории описываемого объекта, тип данных и имя этого объекта.

Например:

```
@deftypevr {Глобальный флаг} int enable
...
@end deftypevr
```

дает в Info следующее:

```
- Глобальный флаг: int enable
...
```

и следующее в печатном руководстве:

**int enable** Глобальный флаг

...

Шаблон такой:

```
@deftypevr категория тип-данных имя
тело-определения
@end deftypevr
```

`@deftypevr` создает для *имени* вхождение в указателе переменных.

`@deftypevar` *тип-данных имя*

Команда `@deftypevar` — это специализированная команда для определений переменных в языках с контролем типов. `@deftypevar` эквивалентна ‘`@deftypevr Переменная ...`’.

Например:

```
@deftypevar int fubar
...
@end deftypevar
```

дает в Info следующее:

```
- Переменная: int fubar
...
```

и следующее в печатном руководстве:

**int fubar** Variable
...

Шаблон такой:

`@deftypevar` *тип-данных имя*
*тело-определения*
`@end deftypevar`

`@deftypevar` создает для *имени* вхождение в указателе переменных.

### 15.4.5 Объектно-ориентированное программирование

Здесь приведены команды для форматирования описаний абстрактных объектов, таких, какие используются в объектно-ориентированном программировании. Класс — это определенный тип абстрактных объектов. Экземпляр класса — это конкретный объект, имеющий тип этого класса. Переменная экземпляра — это переменная, принадлежащая классу, но имеющая в каждом экземпляре свое собственное значение.

В определении, если имя класса на самом деле является именем, определенным для класса в программной системе, вы должны окружать его `@code`, иначе в печатном руководстве оно будет напечатано шрифтом для обычного текста.

`@defcv` *категория класс имя*

Команда `@defcv` — это общая команда для определений переменных, связанных с классами а объектно-ориентированном программировании. После команды `@defcv` следуют три аргумента: категория определяемого, класс, к которому оно принадлежит и его имя. Таким образом,

```
@defcv {Параметр класса} Window border-pattern
...
@end defcv
```

показывает, как вы могли бы написать первую строку определения параметра класса `border-pattern` для класса `Window`.

Шаблон выглядит так:

`@defcv` *категория класс имя*
...
`@end defcv`

`@defcv` создает вхождение в указателе переменных.

`@defivar` *класс имя*

Команда `@defivar` — это команда для определений переменных экземпляров в объектно-ориентированном программировании. `@defivar` эквивалентна ‘`@defcv {Переменная экземпляра} ...`’

Шаблон выглядит так:

```
@defivar класс имя-переменной-экземпляра
тело-определения
@end defivar
```

`@defivar` создает вхождение в указателе переменных.

`@deftypeivar` *класс тип-данных имя*

Команда `@deftypeivar` — это команда для определений типизированных переменных экземпляра в объектно-ориентированном программировании. Она похожа на `@defivar`, но добавляет параметр *тип-данных* для указания типа этой переменной экземпляра. `@deftypeivar` создает вхождение в указателе переменных.

`@defop` *категория класс имя аргументы...*

Команда `@defop` — это общая команда для определений элементов, которые похожи на методы в объектно-ориентированном программировании. Такие элементы принимают аргументы, как функции, но связаны с конкретным классом объектов.

Например, в некоторых системах есть конструкции, называемые *обертками*, которые связаны с классами, как и методы, но действуют во многом подобно макросам, а не как функции. Вы можете использовать `@defop Обертка` для описания одного из них.

Иногда полезно различать методы и *операции*. Вы можете думать об операциях, как об описании для метода. Таким образом, оконная система может указать, что все классы окон имеют метод, называемый `expose`; вы могли бы сказать, что оконная система определяет операцию `expose` для окон вообще. Как правило, у операции есть имя, а также образец аргументов; все методы, реализующие операцию, должны принимать одинаковые аргументы, так как приложения, использующие эту операцию, используют ее, не зная, какой метод ее реализует.

Часто более полезным оказывается документировать операции, а не методы. Например, разработчики оконных приложений должны знать об операции `expose`, но не обязаны быть в курсе, имеет ли данный класс окон собственный метод для реализации этой операции. Для описания этой операции, вы могли бы написать:

```
@defop Операция windows expose
```

Команда `@defop` записывается в начале строки, и за ней на той же строке следуют общее название категории операции, имя класса этой операции, ее имя и аргументы, если таковые имеются.

Вот шаблон:

```
@defop категория класс имя аргументы...
тело-определения
@end defop
```

`@defop` создает вхождение, такое как ‘`expose` для `windows`’, в указателе функций.

`@deftypeop` *категория класс тип-данных имя аргументы...*

Команда `@deftypeop` — это команда определения для типизированных операций в объектно-ориентированном программировании. Она похожа на `@defop`, но в ней лобавлен параметр *тип-данных* для указания типа возвращаемого значения метода. `@deftypeop` создает вхождение в указателе функций.

`@defmethod` *класс имя аргументы...*

Команда `@defmethod` — это команда для определений методов в объектно-ориентированном программировании. Метод — это разновидность функции, которая реализует операцию для конкретного класса объектов и его подклассов. В Лисп-машине методы на самом деле были функциями, но обычно их определяли с помощью `defmethod`.

`@defmethod` эквивалентна '`@defop Метод ...`'. Эта команда записывается в начале строки, и за ней на той же строке следуют имя класса этого метода, имя этого метода и его аргументы, если таковые имеются.

Например,

```
@defmethod bar-class bar-method argument
...
@end defmethod
```

показывает определение метода, называемого `bar-method`, класса `bar-class`. Этот метод принимает один аргумент.

Шаблон таков:

`@defmethod` *класс имя-метода аргументы...*
*тело-определения*
`@end defmethod`

`@defmethod` создает вхождение, такое как '`bar-method` для `bar-class`', в указателе функций.

`@deftypemethod` *класс тип-данных имя аргументы...*

Команда `@deftypemethod` — это команда для определений методов в объектно-ориентированных языках с контролем типов, таких как Си++ и Java. Она похожа на команду `@defmethod`, но в ней добавлен параметр *тип-данных* для задания типа возвращаемого методом значения.

### 15.4.6 Типы данных

Здесь приведены команды для типов данных:

`@deftp` *категория имя атрибуты...*

Команда `@deftp` — это общая команда для типов данных. Эта команда пишется в начале строки, и за ней на той же строке следуют название категории, имя этого типа данных (слово вроде `int` или `float`) и затем имена атрибутов объектов этого типа. Таким образом, вы можете использовать эту команду для описания `int` или `float`, в этом случае вы могли бы использовать в качестве категории слова `тип данных`. (Тип данных является категорией определенных объектов, предназначенной для решения, какие операции можно с ними производить.)

Например, в Лиспе, *парой* называется некий тип данных, и объект этого типа имеет два гнезда, называемых CAR и CDR. Вот как вы могли бы написать первую строку определения для `пары`.

```
@deftp {Тип данных} пара car cdr
...
@end deftp
```

Вот шаблон:

```
@deftp категория имя-типа атрибуты...
тело-определения
@end deftp
```

`@deftp` создает вхождение в указателе типов данных.

## 15.5 Соглашения по написанию определений

Когда вы пишете определение с использованием `@deffn`, `@defun` или одной из других команд для определений, пожалуйста, используйте аргументы, указывающие свой смысл, как аргумент *счетчик* для функции `forward-word`. Также, если имя аргумента содержит имя его типа, позаботьтесь, чтобы аргумент в действительности был этого типа.

## 15.6 Пример определения функции

Определение функции использует команды `@defun` и `@end defun`. Имя этой функции следует сразу после команды `@defun`, а за ним на той же строке пишется список параметров.

Вот определение из раздел “Calling Functions” в *The GNU Emacs Lisp Reference Manual*.

**apply** *function* &rest *arguments* Function

`apply` calls *function* with *arguments*, just like `funcall` but with one difference: the last of *arguments* is a list of arguments to give to *function*, rather than a single argument. We also say that this list is *appended* to the other arguments.

`apply` returns the result of calling *function*. As with `funcall`, *function* must either be a Lisp function or a primitive function; special forms and macros do not make sense in `apply`.

```
(setq f 'list)
     ⇒ list
(apply f 'x 'y 'z)
error  Wrong type argument: 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)
```

An interesting example of using `apply` is found in the description of `mapcar`.

В исходном Texinfo-файле, этот пример выглядит так::

```
@defun apply function &rest arguments
@code{apply} calls @var{function} with
@var{arguments}, just like @code{funcall} but with one
difference: the last of @var{arguments} is a list of
arguments to give to @var{function}, rather than a single
argument.  We also say that this list is @dfn{appended}
to the other arguments.

@code{apply} returns the result of calling
@var{function}.  As with @code{funcall},
@var{function} must either be a Lisp function or a
primitive function; special forms and macros do not make
sense in @code{apply}.

@example
(setq f 'list)
     @result{} list
(apply f 'x 'y 'z)
@error{} Wrong type argument: 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

An interesting example of using @code{apply} is found
in the description of @code{mapcar}.@refill
@end defun
```

В этом руководстве данная функция перечислена в указателе команд и переменных под именем `apply`.

Обычные переменные и пользовательские параметры описываются в похожем формате, за исключением того, что переменные и параметры не принимают аргументов.

# 16 Условно видимый текст

Иногда полезно иметь различающийся текст в разных выходных форматах. Например, вы можете использовать *условные команды*, чтобы указать разный текст для печатного руководства и для Info-файла.

Условные команды не могут быть вложенными.

Условные команды делятся на следующие категории:

- Команды для HTML, Info или TEX.
- Команды не для HTML, Info или TEX.
- Команды непосредственного вызова TEX или HTML.
- Подстановка текста для всех форматов и проверка состояния флага.

## 16.1 Условные команды

Команда `@ifinfo` начинает блок текста, который должен игнорироваться TEX при наборе печатного руководства. Этот блок появится только в Info-файле. Команда `@ifinfo` должна встречаться на отдельной строке; завершайте блок текста, предназначенный только для Info, строкой, содержащей одну команду `@end ifinfo`. В начале Texinfo-файла, разрешения на копирование для Info заключены в блок, помеченный командами `@ifinfo` и `@end ifinfo`. (См. Раздел 3.3 [Обзор для Info и разрешения], с. 34.)

Команды `@iftex` и `@end iftex` похожи на команды `@ifinfo` и `@end ifinfo`, только они задают текст, появляющийся в печатном руководстве, но не в Info-файле. Точно так же команды `@ifhtml` и `@end ifhtml` задают текст, появляющийся только при выводе в формате HTML.

Например,

```
@iftex
Этот текст появится только в печатном руководстве.
@end iftex
@ifinfo
Этот текст появится только в Info.
@end ifinfo
@ifhtml
А этот только в HTML.
@end ifhtml
```

Пример выше выдает следующую строку: Этот текст появится только в печатном руководстве.

Заметьте, что вы видите только одну из двух строк; которую именно, зависит от того, читаете ли вы Info или печатную версию этого руководства.

## 16.2 Отрицательные условные команды

Вы можете пометить фрагмент текста, который должен появляться при выводе во всех форматах, *кроме* некоторого формата, задаваемого одной из следующих `@ifnot...` команд:

```
@ifnothtml ... @end ifnothtml
@ifnotinfo ... @end ifnotinfo
@ifnottex ... @end ifnottex
```

(Команды `@ifnot...` и `@end` на самом деле нужно писать на отдельных строках.)

Если вывод производится не в заданном формате, то блок включается в выходной файл. Иначе он игнорируется.

Блоки, ограниченные этими командами, содержат обычный исходный код Texinfo, как в блоке `@iftex`, а не непосредственные команды программы форматирования, как в блоке `@tex` (см. Раздел 16.3 [Команды прямого форматирования], с. 135).

## 16.3 Непосредственный вызов команд программы форматирования

Внутри области, ограниченной `@iftex` и `@end iftex`, вы можете вставлять некоторые обычные команды TEX. Info игнорирует эти команды, так как они находятся в той части файла, которую читает только TEX. Вы можете писать команды TEX, как вы писали бы их в обычном TEX-файле, но заменяя используемый TEX символ '\' символом '@'. Например, в секции Texinfo-файла `@titlepage` вы можете использовать команду TEX `@vskip` для форматирования страницы с информацией об авторских правах. (Команда `@titlepage` автоматически заставляет Info игнорировать область, точно так же, как команда `@iftex`.)

Однако многие особенности, присущие plain TEX, не будут работать, потому что они перекрываются средствами Texinfo.

Вы можете полностью перейти в режим plain TEX и использовать символ '\' в командах TEX, пометив область командами `@tex` и `@end tex`. (Команда `@tex` также заставляет Info игнорировать область, подобно команде `@iftex`.) Единственное исключение состоит в том, что символ `@` все еще начинает команду, чтобы `@end tex` была правильно распознана.

Вот, для примера, математическое выражение, записанное в формате plain TEX:

```
@tex
$$ \chi^2 = \sum_{i=1}^N
          \left (y_i - (a + b x_i)
          \over \sigma_i\right)^2 $$
@end tex
```

Вывод этого примера появится только в печатном руководстве. Если вы читаете это в Info, вы не увидите уравнения, которое будет выведено в печатном руководстве. В печатном руководстве выражение, написанное выше выглядит так:

$$\chi^2 = \sum_{i=1}^{N} \left( \frac{y_i - (a + bx_i)}{\sigma_i} \right)^2$$

Аналогично, вы можете использовать `@ifhtml ... @end ifhtml` для ограничения области, которую нужно включить только при выводе в формате HTML, и `@html ... @end html` для ограничения области, написанной непосредственно на HTML (опять же, за исключением `@`, так же служащего сигнальным символом, чтобы команда `@end` могла быть распознана.)

## 16.4 `@set`, `@clear` и `@value`

Вы можете указать программам форматирования Texinfo, какие части Texinfo-файла нужно форматировать, а какие пропустить, используя команды `@set`, `@clear`, `@ifset` и `@ifclear`.

Кроме того, вы можете использовать команду `@set` *флаг*, чтобы установить значение *флага* равным строке символов, и команду `@value{`*флаг*`}`, чтобы вставить эту строку в текст. Вы можете использовать `@set`, например, для установки даты и вставлять эту дату в несколько мест Texinfo-файла с помощью `@value`.

### 16.4.1 `@ifset` и `@ifclear`

Когда *флаг* установлен, текст между соответствующими командами `@ifset` *флаг* и `@end ifset` будет отформатирован. Если *флаг* сброшен, программы форматирования Texinfo *не* форматируют текст.

Используйте команду `@set` *флаг* для включения, или *установки флага*; имя *флага* может быть любым одиночным словом, содержащим буквы, цифры, дефисы или подчерки.

Формат команды выглядит так:

```
@set флаг
```

Пишите условно форматируемый текст между командами `@ifset` *флаг* и `@end ifset` следующим образом:

```
@ifset флаг
условный-текст
@end ifset
```

Например, вы можете создать один документ, у которого есть два варианта, скажем, ‘маленькое’ и ‘большое’ руководство:

```
Вы можете использовать эту машину для выкапывания
кустов, не повреждая их.

@set большое

@ifset большое
Она также способна выкапывать взрослые деревья.
@end ifset

Не забудьте быстро пересадить ...
```

В этом примере, текст между `@ifset большое` и `@end ifset` будет отформатирован, потому что флаг `большое` установлен.

Используйте команду `@clear` *флаг* для выключения, или *сброса* флага. Сбрасывание флага противоположно установке. Команда выглядит следующим образом:

```
@clear флаг
```

Пишите эту команду на отдельной строке.

Когда *флаг* сброшен, команды форматирования Texinfo *не* форматируют текст между `@ifset` *флаг* и `@end ifset`; этот текст игнорируется и не появляется ни в печатном руководстве, ни в выводе Info.

Например, если вы сбросили флаг в предыдущем примере, написав команду `@clear большое` после команды `@set большое` (но до условного текста), то команды форматирования Texinfo проигнорируют текст между `@ifset большое` и `@end ifset`. Этот текст не появится в форматированном выводе; и в Info, и в печатном руководстве вы увидите только строки, говорящие “Вы можете использовать эту машину для выкапывания кустов, не повреждая их. Не забудьте быстро пересадить . . .”.

Если флаг сброшен командой `@clear` *флаг*, то текст между соответствующими парами команд `@ifset` *флаг* и `@end ifset` будет отформатирован. Но если флаг установлен с помощью `@set` *флаг*, то команды форматирования *не* форматируют текст между командами `@ifclear` и `@end ifclear`; они игнорируют этот текст. Команда `@ifclear` выглядит следующим образом:

```
@ifclear флаг
```

Кратко, существуют такие команды:

`@set` *флаг*
: Сообщить командам форматирования Texinfo, что *флаг* установлен.

`@clear` *флаг*
: Сообщить командам форматирования Texinfo, что *флаг* сброшен.

`@ifset` *флаг*
: Если *флаг* установлен, предписать командам форматирования Texinfo форматировать текст до следующей команды `@end ifset`.

  Если *флаг* сброшен, предписать командам форматирования Texinfo игнорировать текст до следующей команды `@end ifset`.

`@ifclear` *флаг*
: Если *флаг* установлен, предписать командам форматирования Texinfo игнорировать текст до следующей команды `@end ifclear`.

  Если *флаг* сброшен, предписать командам форматирования Texinfo игнорировать текст до следующей команды `@end ifclear`.

### 16.4.2 `@set` и `@value`

Вы можете использовать команду `@set` для установки значения флага, которое потом может быть получено командой `@value`. Флаг — это идентификатор; для наилучших результатов используйте в имени флага только буквы и цифры, но не ‘`-`’ или ‘`_`’ — они сработают в некоторых контекстах, но не всегда, из-за ограничений в TEX. Значение — это просто цепочка знаков, остаток строки ввода.

Команда `@set` записывается подобным образом:

```
@set foo Это строка.
```

Этот пример устанавливает значение флага `foo` равным “Это строка.”.

Программы форматирования Texinfo замещают команду `@value{`*флаг*`}` строкой, в значение которой установлен *флаг*. Таким образом, если `foo` установлен, как показано выше, то программы форматирования Texinfo преобразуют

```
@value{foo}
```

в строку

```
Это строка.
```

Вы можете писать команду `@value` внутри абзаца, но команду `@set` вы должны писать на отдельной строке.

Если вы напишите команду `@set` следующим образом:

```
@set foo
```

не задавая строку, то значением `foo` будет пустая строка.

Если вы очищаете ранее установленный флаг командой `@clear` *флаг*, последующая команда `@value{флаг}` будет неверна, и строка заменяется сообщением об ошибке: '`{Значение "`*флаг*`" не задано}`'.

Например, если вы установили `foo` так:

```
@set насколько очень, очень, очень
```

то программы форматирования преобразуют

```
Сегодня @value{насколько} сырой день.
```

в строку

```
Сегодня очень, очень, очень сырой день.
```

Если вы напишите

```
@clear насколько
```

то программы форматирования преобразуют

```
Сегодня @value{насколько} сырой день.
```

в строку

```
Сегодня {Значение "насколько" не задано} сырой день.
```

### 16.4.3 Пример применения `@value`

Вы можете использовать команду `@value`, чтобы уменьшить количество мест в тексте, которые вам нужно изменить при внесении в руководство исправлений или дополнений. Здесь показано, как это сделано в *Руководстве по GNU Make*:

1. Установлены флаги:

   ```
   @set EDITION 0.35 Beta
   @set VERSION 3.63 Beta
   @set UPDATED 14 августа 1992
   @set UPDATE-MONTH август 1992
   ```

2. Написан текст для первой секции `@ifinfo`, для тех, кто читает Texinfo-файл:

   ```
   Это редакция @value{EDITION},
   последние исправления @value{UPDATED},
   @cite{Руководства по GNU Make}
   для @code{make} версии @value{VERSION}.
   ```

3. Написан текст для титульного листа, для читающих печатное руководство:

   ```
   @title GNU Make
   @subtitle Программа управления перекомпиляцией
   @subtitle Редакция @value{EDITION}, ...
   @subtitle @value{UPDATE-MONTH}
   ```

   (На обложке книги, дата, сообщающая месяц и год выпуска, смотрится более уместно, чем показывающая кроме этого также и число.)

4. Написан текст для первой ноды, для людей, читающих Info-файл:

```
Это редакция @value{EDITION},
последние исправления @value{UPDATED},
@cite{Руководства по GNU Make}
для @code{make} версии @value{VERSION}.
```

После форматирования руководства текст в первой секции `@ifinfo` выглядит следующим образом:

```
Это редакция 0.35 Beta ‘Руководства по GNU Make’,
последние исправления 14 августа 1992,
для ‘make’ версии 3.63 Beta.
```

Когда вы исправляете руководство, измените только значения флагов; нет необходимости переписывать три секции.

# 17 Поддержка разных языков

В Texinfo есть некоторая поддержка для документов, написанных не на английском языке, хотя в этой области еще предстоит сделать значительную работу.

Для получения перечня различных акцентов и специальных символов, поддерживаемых Texinfo, смотрите Раздел 13.3 [Вставка акцентов], с. 108.

## 17.1 `@documentlanguage` *cc*: Задание языка документа

Команда `@documentlanguage` объявляет язык текущего документа. Пишите ее на отдельной строке, после нее пишите двухсимвольный код языка ISO-639 (перечень ниже). Если у вас есть документ на нескольких языках, предполагается, что вы можете использовать ее несколько раз, для объявления каждого изменения языка. Если эта команда не написана, по умолчанию используется значение `en`, английский язык.

В настоящее время эта команда игнорируется при выводе в Info и HTML. В TEX, она считывает файл ‘`txi-`*cc*`.tex`’ (если он существует). Эти файлы переопределяют различные английские слова, используемые в выводе TEX, такие как ‘Chapter’, ‘See’ и так далее.

Было бы неплохо, если бы эти команды изменяли также представление TEX о текущих образцах переносов (с помощью примитива TEX `\language`), но, к сожалению, пока это не реализовано.

## 17.2 `@documentencoding` *enc*: Задание входной кодировки

Команда `@documentencoding` объявляет входную кодировку документа. Пишите ее на отдельной строке, после нее пишите правильное описание кодировки, такое как ‘`ISO-8859-1`’.

На данный момент это используется только при выводе `makeinfo` в формате HTML. Если задана кодировка документа *код*, она используется в теге ‘`<meta>`’ в секции ‘`<head>`’:

```
<meta http-equiv="Content-Type" content="text/html; charset=код">
```

# 18 Определение новых команд Texinfo

Texinfo предоставляет несколько способов определить новые команды:

- Команда Texinfo *macro* позволяет вам определять новые команды в виде виде произвольной последовательности текста и/или существующих команд Texinfo (включая другие макросы). Макрос может принимать любое число *параметров* — фрагментов текста, которые вы передаете при каждом вызове макроса.

  Кстати, такие макросы не имеют никакого отношения к команде `@defmac`, которая служит для документирования рассматриваемых в руководстве макросов (см. Раздел 15.1 [Шаблон определения], с. 121.)
- '`@alias`' дает удобный способ определения нового имени для существующей команды.
- '`@definfoenclose`' позволяет вам определять новые команды с настраиваемым видом при выводе в Info.

## 18.1 Определение макросов

Используйте для определения макросов команду Texinfo `@macro`, как показано:

```
@macro имя-макроса{парам1, парам2, ...}
текст ... \парам1\ ...
@end macro
```

*Параметры парам1*, *парам2*, ... соответствуют аргументам, передаваемым макросу при последующих вызовах внутри документа (описано в следующем разделе).

Чтобы макрос работал в TEX, *имя-макроса* должно состоять только из букв и не включать цифр, подчерков, дефисов или других специальных знаков.

Если макрос не нуждается в параметрах, вы можете определить его с пустым списком параметров ('`@macro foo {}`') или вообще без фигурных скобок ('`@macro foo`').

В определении *тела* макроса может содержаться большинство команд Texinfo, в том числе и определенные ранее макросы. Вызовы еще не определенных макросов запрещены; таким образом, невозможно создать взаимно рекурсивные макросы Texinfo. Кроме того, определение макроса, определяющего другой макрос не работает в TEX из-за ограничений в реализации `@macro`.

В теле макроса, вхождения имен параметров, заключенных в символы обратной косой черты, как например '*\парам1*' в примере выше, замещаются соответствующим аргументом при вызове. Вы можете использовать в теле имена параметров любое число раз, включая ноль.

Чтобы получить в раскрытии макроса один знак '\', используйте '\\'. Любое другое применение '\' в теле порождает предупреждение.

Переводы строк после строки `@macro` и перед строкой `@end macro` игнорируются, то есть не включаются в тело макроса. Все остальные пропуски интерпретируются в соответствии с обычными правилами Texinfo.

Чтобы позволить макросу использоваться рекурсивно, то есть в аргументе вызова самого себя, вы должны определить его с помощью '`@rmacro`', как показано:

```
@rmacro rmac
a\arg\b
@end rmacro
...
@rmac{1@rmac{text}2}
```

Это дает на выходе ‘a1atextb2b’. При использовании ‘`@macro`’ вместо ‘`@rmacro`’, будет выдано сообщение об ошибке.

Вы можете отменить определение макроса *foo* вызовом `@unmacro` *foo*. Отмена еще не определенного макроса не является ошибкой. Например:

```
@unmacro foo
```

## 18.2 Вызов макросов

После того как макрос определен (смотрите предыдущий раздел), вы можете использовать (*вызывать*) его в ваших документах следующим образом:

`@`*имя-макроса* `{`*арг1*`, `*арг2*`, ...}`

в результате получится так, как будто вы напечатали тело макроса *имя-макроса* в этом месте. Например:

```
@macro foo {p, q}
Вместе: \p\ и \q\.
@end macro
@foo{А, Б}
```

дает:

Вместе: А и Б.

Итак, аргументы и параметры разделяются запятыми и заключаются в фигурные скобки; любые пробельные символы после (но не перед) запятой игнорируются. При вызове (но не в определении) неоходимо писать фигурные скобки, даже если макро не принимает аргументов, так же, как и во всех остальных командах Texinfo. Например:

```
@macro без-аргументов {}
Здесь нет аргументов.
@end macro
@без-аргументов{}
```

дает:

Здесь нет аргументов.

Чтобы вставить в аргумент запятую, фигурную скобку или обратную косую черту, напишите перед ней символ обратной косой черты, например

`@`*имя-макроса* `{\\\{\}\,}`

передаст макросу *имя-макроса* аргумент ‘`\{},`’, (что почти наверняка приведет к ошибке).

Если макрос определен с одним аргументом и вызван без фигурных скобок, в качестве аргумента ему будет передан весь остаток строки после имени макроса. Например:

```
@macro bar {p}
Дважды: \p\ и \p\.
@end macro
@bar ax
```

дает:

Twice: ax и ax.

Если макрос определен с одним аргументом и вызван с фигурными скобками, в качестве аргумента передается текст в скобках, независимо от запятых. К примеру:

```
@macro bar {p}
Twice: \p\ и \p\.
@end macro
@bar{a,б}
```

дает:

Дважды: a,б и a,б.

## 18.3 Подробно о макросах

В связи с неразрешимыми разногласиями в реализациях TEX и `makeinfo` макросы Texinfo обладают следующими ограничениями.

- Все макросы раскрываются внутри по меньшей мере одной группы TEX. Это означает, что `@set` и другие команды не будет действовать внутри макроса.
- Макросы, содержащие команду, которая должна находиться на отдельной строке, например условие, нельзя вызывать в середине строки.
- Реализация для TEX не умеет конструировать макросы, которые определяют другие макросы естественным способом. Чтобы сделать сделать это, вы обязаны использовать условные команды и непосредственно TEX. Например:

```
@ifinfo
@macro ctor {name, arg}
@macro \name\
нечто подразумевающее \arg\
@end macro
@end macro
@end ifinfo
@tex
\gdef\ctor#1{\ctorx#1,}
\gdef\ctorx#1,#2,{\def#1{нечто подразумевающее #2}}
@end tex
```

- Лучше избегать комментариев в определениях макросов.

## 18.4 ‘`@alias` *новая=существующая*’

Команда ‘`@alias`’ определяет новую команду, которая действует точно так же, как существующая. Это полезно для определения дополнительных разметочных имен, которые сохранят семантическую информацию на входе, хотя и, возможно, оставят результат таким же.

Пишите команду ‘@alias’ на отдельной строке, а после пишите имя новой команды, знак равенства и имя существующей команды. Пропуски вокруг знака равенства игнорируются. Пример:

```
@alias новая = существующая
```

Например, если ваш документ содержит ссылки как на книги, так и на другие произведения (скажем, фильмы), вы могли бы захотеть определить макро @moviecite{}, которое делает то же самое, что и обычная команда @cite{}, но также передает дополнительную семантическую информацию. Вы могли бы сделать это так:

```
@alias moviecite = cite
```

Макросы не всегда имеют тот же эффект из-за капризов разбора аргументов. Кроме того, определять псевдонимы намного проще, чем макросы. Так что эта команда не избыточна. (Она также интенсивно применялась в Jargon File!)

Псевдонимы не должны быть рекурсивными, прямо или косвенно.

## 18.5 ‘definfoenclose’: Настройка выделения

Вы можете использовать обычные команды TEX внутри блока @iftex ... @end iftex для создания ваших собственных выделяющих команд для Texinfo. Самый простой способ достичь этого — приравнять ваши команды уже существующим, например командам, выводящим курсивом. Такие новые команды работают только в TEX.

Вы можете использовать команду @definfoenclose внутри блока @ifinfo ... @end ifinfo для определения новых команд для Info с теми же именами, что и новые команды для TEX. @definfoenclose создает новые команды для Info, которые помечают текст, окружая его заданными строками.[1]

Ниже показано, как создать новую @-команду, называемую @phoo, которая заставляет TEX набирать свой аргумент курсивом, а Info — выводить аргумент между ‘//’ и ‘\\’.

Для TEX напишите следующее, чтобы приравнять команду @phoo существующей команде @i для курсива:

```
@iftex
@global@let@phoo=@i
@end iftex
```

Это определяет @phoo как команду, заставляющую TEX набирать аргумент @phoo курсивом. @global@let говорит TEX приравнять следующий аргумент аргументу, идущему после знака равенства.

Для Info напишите следующее, чтобы программы форматирования Info заключали аргумент между ‘//’ и ‘\\’:

```
@ifinfo
@definfoenclose phoo,//,\\
@end ifinfo
```

[1] На данный момент @definfoenclose работает только с texinfo-format-buffer и texinfo-format-region, но не с makeinfo.

Пишите команду `@definfoenclose` в начале строки, а после нее пишите три аргумента, разделенные запятыми (запятые используются в качестве разделителей в строке `@node` таким же образом).

- Первым аргументом `@definfoenclose` является имя `@`-команды **без** ‘`@`’;
- второй аргумент — это начальная строка-разделитель для Info; и
- третий аргумент — завершающая строка-разделитель.

Два последних аргумента окружают выделяемый текст в Info-файле. Строка-разделитель может содержать пробельные знаки, но они имеют значение, поэтому не вставляйте пропуски, если не хотите, чтобы они появились при выводе. Ни начальный, ни завершающий разделитель не обязателен. Однако, если вы не задаете начальный разделитель, вы должны написать после имени команды две запятые подряд; иначе команды форматирования для Info неправильно воспримут завершающий разделитель как начальный.

После того, как вы определили `@phoo` и для TEX, и для Info, вы можете написать `@phoo{bar}` и получить ‘`//bar\\`’ в Info и курсивное *bar* в печатном выводе.

Заметьте, что каждое определение применяется для своей форматирующей программы: одно для TEX, другое для Info.

Вот другой пример:

```
@ifinfo
@definfoenclose headword■:
@end ifinfo
@iftex
@global@let@headword=@b
@end iftex
```

Это определяет `@headword` как команду, которая в Info вставляет двоеточие после аргумента и не вставляет ничего перед ним, а в TEX набирает аргумент жирным шрифтом.

# 19 Форматирование и печать твердой копии

Есть три основные команды оболочки для создания печатного руководства из Texinfo-файла: одна для преобразования Texinfo-файла в печатаемый файл, вторая для сортировки именных указателей и третья для печати отформатированного документа. Когда вы используете эти команды, вы можете работать непосредственно из оболочки операционной системы или из оболочки внутри GNU Emacs.

Если вы пользуетесь GNU Emacs, вы можете использовать вместо команд оболочки команды, предоставляемые режимом Texinfo. Кроме трех команд для форматирования файла, сортировки именных указателей и печати результата, режим Texinfo предоставляет привязки ключей для команд центрирования буфера вывода, показа очереди печати и удаления задания из очереди печати.

## 19.1 Используйте TEX

Для форматирования Texinfo-файлов используется программа для набора, называемая TEX. TEX — это очень мощная программа подготовки печатных документов и, если ее правильно использовать, работает исключительно хорошо. (См. Приложение J [Как получить TEX], с. 221, для информации о том, как получить TEX.)

Команды `makeinfo`, `texinfo-format-region` и `texinfo-format-buffer` читают в Texinfo-файле те же самые `@`-команды, что и TEX, но обрабатывают их иначе для создания Info-файла; смотрите Раздел 20.1 [Создание Info-файла], с. 158.

## 19.2 Форматирование с помощью `tex` и `texindex`

Форматируйте Texinfo-файл с помощью команды оболочки `tex`, за которой стоит имя Texinfo-файла. Например:

```
tex foo.texi
```

TEX создаст *DVI-файл*, а также несколько вспомогательных файлов, содержащих сведения об именных указателях, перекрестных ссылках и другие. DVI-файл (от *DeVice Independent*, то есть независимый от устройства) можно напечатать практически на любом виде печатающих устройств (смотрите следующие разделы).

Форматирующая команда `tex` сама по себе не сортирует именные указатели, она записывает файл с несортированными данными для указателей. (Команда `texi2dvi` автоматически создает именные указатели; см. Раздел 19.3 [Форматирование с помощью `texi2dvi`], с. 148.) Чтобы создать печатный именной указатель после прогона команды `tex`, сначала вам понадобится для работы сортированный указатель. Команда `texindex` сортирует именные указатели. (Исходный файл ‘`texindex.c`’ поставляется как часть стандартного дистрибутива Texinfo, кроме того, его можно найти в других местах.)

Форматирующая команда `tex` выдает несортированные файлы именных указателей с именами, подчиняющимися стандартному соглашению: имя вашего главного входного файла с удаленным расширением ‘`.tex`’ (или другим, см. раздел “tex invocation” в *Web2c*), за которым следуют две буквы имени указателя. Например, необработанные выходные файлы с именными указателями для входного файла ‘`foo.texinfo`’

назывались бы ‘`foo.cp`’, ‘`foo.vr`’, ‘`foo.fn`’, ‘`foo.tp`’, ‘`foo.pg`’ и ‘`foo.ky`’. Это в точности те аргументы, которые нужно передать `texindex`.

Вместо явного задания всех файлов с несортированными именными указателями, вы можете использовать ‘`??`’ в качестве шаблона оболочки и дать команду в такой форме:

```
texindex foo.??
```

Эта команда запустит `texindex` для всех файлов с несортированными указателями, включая те, которые вы определили сами с помощью `@defindex` или `@defcodeindex`. (Вы также можете выполнить ‘`texindex foo.??`’, даже если существуют файлы, называемые похожим образом, с двухбуквенным расширением, такие как ‘`foo.el`’. Команда `texindex` сообщает о таких файлах, но игнорирует их.)

Для каждого заданного файла `texindex` создает файл с сортированным именным указателем, имя которого получается добавлением ‘`s`’ к концу имени входного файла. Команда `@printindex` ищет файлы с такими именами (см. Раздел 4.1 [Печать именных указателей и меню], с. 43). `texindex` не изменяет исходный файл.

После того, как вы отсортировали именные указатели, вам нужно снова запустить форматирующую команду `tex` для Texinfo-файла. Это создаст DVI-файл, на этот раз с соответствующими действительности вхождениями именных указателей.

Наконец, вам может понадобиться запустить `tex` еще один раз, чтобы получить правильные номера страниц в перекрестных ссылках.

Кратко, вот процесс из пяти шагов:

1. Запустите `tex` для вашего Texinfo-файла. Это создаст DVI-файл (с неопределенными перекрестными ссылками и без именных указателей) и исходные файлы (с двухбуквенными расширениями) с именными указателями.
2. Запустите `texindex` для исходных файлов с именными указателями. Это создаст соответствующие файлы (с трехбуквенными расширениями) с сортированными указателями.
3. Снова запустите `tex` для вашего Texinfo-файла. Это заново создаст DVI-файл, на этот раз с именными указателями и определенными перекрестными ссылками, но номера страниц для перекрестных ссылок, оставшиеся с последнего раза, как правило неверны.
4. Опять отсортировать именные указатели с помощью `texindex`.
5. Запустите `tex` последний раз. На этот раз для перекрестных ссылок записываются правильные номера страниц.

Или это процесс из одного фага: запустите `texi2dvi` (см. Раздел 19.3 [Форматирование с texi2dvi], с. 148).

Вам не нужно запускать `texindex` каждый раз после запуска `tex`. Если вы не этого не сделали, то при следующем запуске форматирующая команда `tex` будет использовать файлы с сортированными именными указателями, оставшиеся от прошлой работы `texindex`. Обычно это подходит в процессе отладки.

Иногда вы можете захотеть напечатать документ, хотя знаете, что он неполный, или напечатать только одну главу документа. В таком случае обычные вспомогательные файлы, которые создает TEX, и предупреждения, которые он выдает, когда

перекрестные ссылки неверны, только мешают. Вы можете избежать их появления с помощью команды `@novalidate`, которую вы должны дать *до* команды `@setfilename` (см. Раздел 3.2.3 [@setfilename], с. 30). Таким образом, начало вашего файла могло бы выглядеть примерно так:

```
\input texinfo
@novalidate
@setfilename myfile.info
...
```

`@novalidate` также выключает проверку в `makeinfo`, как ее ключ `--no-validate` (см. Раздел 20.1.4 [Проверка указателей], с. 162).

## 19.3 Форматирование с помощью `texi2dvi`

Команда `texi2dvi` автоматически запускает `tex` и `texindex` столько раз, сколько необходимо для создания DVI-файла с сортированными именными указателями и всеми разрешенными перекрестными ссылками. Она упрощает последовательность `tex` — `texindex` — `tex` — `tex`, описанную в предыдущем разделе.

Чтобы запустить `texi2dvi` для входного файла ‘`foo.texi`’, сделайте следующее (где ‘`prompt$ `’ это приглашение вашей оболочки):

```
prompt$ texi2dvi foo.texi
```

Как показано в этом примере, имена входных файлов для `texi2dvi` должны включать любое расширение (‘`.texi`’, ‘`.texinfo`’, etc.). Под MS-DOS и, возможно, в других обстоятельствах вам может понадобиться запускать ‘`sh texi2dvi foo.texi`’, а не полагаться на то, что операционная система запустит для сценария ‘`texi2dvi`’ оболочку.

Пожалуй, наиболее полезный ключ для `texi2dvi` — это ‘`-texinfo=`*команда*’. Он вставляет *команду* на отдельной строке после `@setfilename` во временной копии входного файла перед запуском TEX. С ним, вы можете задать различные форматы печати, такие как `@smallbook` (см. Раздел 19.11 [smallbook], с. 154), `@afourpaper` (см. Раздел 19.12 [Формат A4], с. 155) или `@pageparams` (см. Раздел 19.13 [pagesizes], с. 155), не меняя в действительности исходный текст документа. (Вы также можете сделать это для всей системы с помощью ‘`texinfo.cnf`’; см. Раздел 19.9 [Подготовка к применению TEX], с. 152).

Для получения списка ключей запустите ‘`texi2dvi -help`’.

## 19.4 Печать в оболочке с помощью `lpr -d`

Точная команда для печати DVI-файла зависит от вашей системы, но обычно это ‘`lpr -d`’. Эта команда принимает имя DVI-файла без расширения или с расширением ‘`.dvi`’. (Если это команда ‘`lpr`’, вы должны включать ‘`.dvi`’.)

Следующих команд, например, будет (может быть) достаточно для сортировки именных указателей, форматирования и печати *Руководства по Bison*:

```
tex bison.texinfo
texindex bison.??
tex bison.texinfo
lpr -d bison.dvi
```

(Помните, что команды оболочки могут быть другими в вашей системе; но это чаще всего используемые версии.)

При использовании сценария оболочки `texi2dvi`, вам нужно просто напечатать:

```
texi2dvi bison.texinfo
lpr -d bison.dvi
```

`lpr` — это стандартная программа в системах Unix, но она обычно отсутствует в MS-DOS/MS-Windows. Некоторые сетевые пакеты могут поставляться с программой, называемой `lpr`, но их возможности обычно ограничены посылкой файлов по сети на сервер печати, и обычно они не поддерживают ключ '`-d`'. Если вы насколько невезучи, что работаете на одной из этих систем, у вас есть несколько альтернативных способов напечатать DVI-файлы:

- Найдите и установите Unix-подобную программу `lpr` или ее имитацию. Если вы можете это сделать, то сможете печатать DVI-файлы точно так, как описано выше.
- Пошлите DVI-файлы в очередь DVI-файлов сетевого принтера. Некоторые сетевые принтеры имеют специальные очереди для печати DVI-файлов. Вы наверняка можете настроить сетевое программное обеспечение для посылки файлов в эту очередь. В некоторых случаях версия `lpr`, поставляемая с вашим сетевым программным обеспечением, имеет специальные ключи для посылки файла в конкретную очередь, как здесь:

  ```
  lpr -Qdvi -hprint.server.domain bison.dvi
  ```

- Преобразуйте DVI-файл в Postscript- или PCL-файл и пошлите его на ваш локальный принтер. См. раздел "dvips invocation" в *Dvips*, страницы man для `dvilj` для подробного описания этих инструментов. Когда DVI-файл преобразован в формат, который ваш принтер понимает непосредственно, вы можете послать его на соответствующий порт, как правило это '`PRN`'.

## 19.5 Из оболочки Emacs

Вы можете исполнить команды форматирования и печати из подчиненной оболочки GNU Emacs. Чтобы создать в Emacs оболочку, напечатайте *`M-x shell`*. В этой оболочке вы можете форматировать и печатать документ. См. Глава 19 [Форматирование и печать твердой копии], с. 146, для подробностей.

Вы можете переключиться в буфер оболочки и из него во время работы `tex` и редактировать что-то еще. Если вы форматируете длинный документ на медленной машине, это может быть очень удобно.

Вы можете также использовать `texi2dvi` из оболочки Emacs. Например, так можно применить `texi2dvi` для форматирования и печати книги *Использование и перенос GNU CC* из оболочки в Emacs:

```
texi2dvi gcc.texinfo
lpr -d gcc.dvi
```

## 19.6 Форматирование и печать в режиме Texinfo

Режим Texinfo предоставляет несколько предопределенных команд, привязанных к ключам, для форматирования и печати с TEX. Они включают команды для сортировки именных указателей, просмотра очереди печати, уничтожения форматирующего задания и центрирования буфера, в котором происходят эти действия.

*C-c C-t C-b*
*M-x texinfo-tex-buffer*
: Запускает `texi2dvi` для текущего буфера.

*C-c C-t C-r*
*M-x texinfo-tex-region*
: Запускает TEX для текущего буфера.

*C-c C-t C-i*
*M-x texinfo-texindex*
: Сортирует именные указатели Texinfo-файла, отформатированного с помощью `texinfo-tex-region`.

*C-c C-t C-p*
*M-x texinfo-tex-print*
: Печатает DVI-файл, отформатированный с помощью `texinfo-tex-region` или `texinfo-tex-buffer`.

*C-c C-t C-q*
*M-x tex-show-print-queue*
: Показывает очередь печати.

*C-c C-t C-d*
*M-x texinfo-delete-from-print-queue*
: Удаляет задание из очереди печати; у вас запросят номер задания, показанный ранее командой *C-c C-t C-q* (`texinfo-show-tex-print-queue`).

*C-c C-t C-k*
*M-x tex-kill-job*
: Уничтожает работающее в данный момент задание TEX, запущенное с помощью `texinfo-tex-region` или `texinfo-tex-buffer`, или любой другой процесс, работающий в буфере оболочки Texinfo.

*C-c C-t C-x*
*M-x texinfo-quit-job*
: Прекращает форматирующее задание TEX, которое было остановлено из-за ошибки, послав ему (x). Когда вы делаете это, TEX сохраняет запись сделанных действий в '`.log`'-файле.

*C-c C-t C-l*
*M-x tex-recenter-output-buffer*
: Перерисовать буфер оболочки, в котором запущены форматирующие и печатающие команды TEX, для показа последних строк вывода.

Таким образом, обычная последовательность команд для форматирования буфера выглядит, как показано ниже (с комментариями справа):

```
C-c C-t C-b     Запустить texi2dvi для буфера.
C-c C-t C-p     Напечатать DVI-файл.
C-c C-t C-q     Показать очередь принтера.
```

Команды форматирования с TEX в режиме Texinfo запускают в Emacs подоболочку, называемую '`*tex-shell*`'. Команды `texinfo-tex-command`, `texinfo-texindex-command` и `tex-dvi-print-command` работают в этой оболочке.

Вы можете наблюдать за работой команд в буфере '`*tex-shell*`', переключаться в него и из него и использовать буфер '`*tex-shell*`' как любой другой буфер оболочки.

Команды печати и форматирования зависят от нескольких переменных. Их значения по умолчанию:

| Переменная | Значение по умолчанию |
|---|---|
| `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"` |

Вы можете изменить значения этих переменных с помощью команды *`M-x edit-options`* (см. раздел "Editing Variable Values" в *Руководство по GNU Emacs*), команды *`M-x set-variable`* (см. раздел "Examining and Setting Variables" в *Руководство по GNU Emacs*) или вашего файла инициализации '`.emacs`' (см. раздел "Init File" в *Руководство по GNU Emacs*).

Начиная с версии 20, GNU Emacs предоставляет дружественный интерфейс, называемый *Customize*, для изменения значений переменных, задаваемых пользователем. См. раздел "Easy Customization Interface" в *Руководство по GNU Emacs*, для подробностей об этом пакете. Переменные Texinfo можно найти в группе '`Development/Docs/Texinfo`', когда вы вызвали команду *`M-x customize`*.

## 19.7 Использование списка локальных переменных

Еще один способ применить команду форматирования с TEX к Texinfo-файлу — поместить эту команду в *список локальных переменных* в конце этого Texinfo-файла. Вы можете задать команду `tex` или `texi2dvi` в качестве переменной `compile-command` и велеть Emacs запустить ее, напечатав *`M-x compile`*. Это создаст специальную оболочку, называемую буфером '`*compilation*`', в которой Emacs запускает команду компиляции. Например, в конце файла '`gdb.texinfo`', после `@bye`, вы могли бы написать следующее:

```
Local Variables:
compile-command: "texi2dvi gdb.texinfo"
End:
```

Этот метод чаще всего применяется программистами, которые также компилируют таким образом программы; смотрите раздел “Compilation” в *Руководство по GNU Emacs*.

## 19.8 Обзор необходимого для форматирования с TEX

Каждый Texinfo-файл, предназначенный для обработки TEX, должен начинаться командой `\input` и содержать команду `@setfilename`:

```
\input texinfo
@setfilename arg-not-used-by-TeX
```

Первая команда указывает TEX загрузить макросы, которые нужны ему для обработки Texinfo-файла, а вторая команда открывает вспомогательные файлы.

Каждый Texinfo-файл должен заканчиваться строкой, прекращающей работу TEX и выводящей незавершенные страницы:

```
@bye
```

Строго говоря, эти строки — все, что нужно, чтобы Texinfo-файл был успешно обработан TEX.

Однако, обычно начало включает команду `@settitle` для определения названия печатного руководства, команду `@setchapternewpage`, титульный лист, страницу с информацией об авторских правах и разрешения на копирование. Кроме `@bye`, конец файла обычно включает именные указатели и содержание. (И конечно, большинство руководств также содержат тело текста.)

Для дальнейшей информации смотрите:

- Раздел 3.2.4 [`@settitle`], с. 31,
- Раздел 3.2.5 [`@setchapternewpage`], с. 32,
- Приложение F [Заголовки страниц], с. 206,
- Раздел 3.4 [Титульный лист], с. 35,
- Раздел 4.1 [Печать именных указателей и меню], с. 43, а также
- Раздел 4.2 [Содержание], с. 44.

## 19.9 Подготовка к применению TEX

TEX должен знать, где найти файл ‘`texinfo.tex`’, который вы велели ему включить командой ‘`\input texinfo`’ в начале первой строки. Файл ‘`texinfo.tex`’ говорит TEX, как обращаться с `@`-командами; он включается во все стандартные дистрибутивы GNU.

Обычно файл ‘`texinfo.tex`’ помещается в каталог по умолчанию, который содержит макросы TEX, когда устанавливается GNU Emacs или другое программное обеспечение GNU. (По умолчанию это ‘`/usr/local/share/texmf/tex/texinfo/texinfo.tex`’.) В этом случае TEX сможет найти файл, и вам не понадобится делать ничего особенного. Или вы можете поместить ‘`texinfo.tex`’ в текущий каталог при запуске TEX, и TEX найдет его там.

Также, вам нужно установить ‘`epsf.tex`’ в то же место, что и ‘`texinfo.tex`’, если он не был уже установлен из другого дистрибутива. Этот файл нужен для поддержки команды `@image` (см. Раздел 13.11 [Рисунки], с. 115).

При желании вы можете создать дополнительный файл ‘`texinfo.cnf`’ и установить и его. TEX читает этот файл во время исполнения команды `@setfilename` (см. Раздел 3.2.3 [@setfilename], с. 30). Вы можете поместить в нем любые команды по вашему желанию, в соответствии с локальными общесистемными соглашениями. Они будут читаться TEX при обработке любого документа Texinfo. Например, если ‘`texinfo.cnf`’ содержит строку ‘`@afourpaper`’ (см. Раздел 19.12 [Формат A4], с. 155), то все документы Texinfo будут обрабатываться с таким размером страницы. Если вам нечего написать в ‘`texinfo.cnf`’, вам не обязательно его создавать.

Если ни одно из указанных выше положений этих системных файлов для вас не достаточно, вы можете задать каталоги явно. Для ‘`texinfo.tex`’, вы можете сделать это, написав полный путь к файлу после команды `\input`. Другой способ, работающий и для ‘`texinfo.tex`’, и для ‘`texinfo.cnf`’ (и любого другого файла, которой мог бы читать TEX), — установить переменную среды `TEXINPUTS` в вашем файле ‘`.cshrc`’ или ‘`.profile`’.

Какой из ‘`.cshrc`’ или ‘`.profile`’ вам нужен, зависит от того, используете ли вы совместимый с Bourne shell (`sh`, `bash`, `ksh`, . . . ) или совместимый с C shell (`csh`, `tcsh`) командный интерпретатор. Последний читает для инициализации файл ‘`.cshrc`’, а первый читает ‘`.profile`’.

В файле ‘`.cshrc`’, вы можете использовать следующую последовательность команд `csh`:

```
setenv TEXINPUTS .:/home/me/mylib:/usr/lib/tex/macros
```

В файле ‘`.profile`’, вы можете использовать следующую последовательность команд `sh`:

```
TEXINPUTS=.:/home/me/mylib:/usr/lib/tex/macros
export TEXINPUTS
```

В MS-DOS/MS-Windows, вы могли бы сказать это таким образом[1]:

```
set TEXINPUTS=.;d:/home/me/mylib;c:/usr/lib/tex/macros
```

Обычно пользователи DOS/Windows помещают такие команды в файл ‘`autoexec.bat`’ или Реестр Windows.

Эти установки заставили бы TEX искать файл для ‘`\input`’ сначала в текущем каталоге, обозначаемом ‘`.`’, затем в каталоге гипотетического пользователя ‘`me/mylib`’ и, наконец, в системном каталоге ‘`/usr/lib/tex/macros`’.

Наконец, вы можете захотеть сделать дамп форматного файла (см. раздел “Memory dumps” в *Web2c*), чтобы TEX мог загружать Texinfo быстрее. (Недостаток этого в том, что при обновлении ‘`texinfo.tex`’ потребуется повторный дамп.) Вы можете сделать это, запустив такую команду, в предположении, что TEX находит ‘`epsf.tex`’:

```
initex texinfo @dump
```

[1] Обратите внимание на использование в качестве разделителя каталогов символа ‘`;`’ вместо ‘`:`’ в этих системах.

(`@dump` — это примитив TEX.) Затем вам нужно перенести ‘`texinfo.fmt`’ в то место, где находятся ваши `.fmt`-файлы; обычно это подкаталог ‘`web2c`’ вашего TEX, например, ‘`/usr/local/share/tex/web2c`’.

## 19.10 Переполненные боксы

Иногда TEX не может набрать строку, не расширяя ее за правый край. Это может случиться, когда TEX встречает что-то, что он интерпретирует как длинное слово, которое он не может перенести, такое как адрес электронной почты или очень длинный заголовок. Когда такое случается, TEX печатает сообщение об ошибке вроде этого:

```
Overfull @hbox (20.76302pt too wide)
```

(В TEX, строки являются “горизонтальными боксами”, отсюда термин “hbox”. ‘`@hbox`’ — это примитив TEX, не нужный в языке Texinfo.)

TEX также выдает номер строки в исходном Texinfo-файле и текст плохой строки, который помечен во всех местах, которые TEX счел точками возможного переноса. См. Раздел G.2 [Поиск ошибок с TEX], с. 212, для большей информации об ошибках при наборе.

Если в Texinfo-файле есть переполненный горизонтальный бокс, вы можете переписать предложение так, чтобы бокс не переполнялся, или вы можете решить оставить его. Небольшое вторжение на правое поле часто не играет роли и даже может быть незаметно.

Если у вас есть много переполненных боксов и/или антипатия к переписыванию, вы можете увеличить допустимый междусловный пропуск, избегнув таким образом (если повезет) много неудачных разрывов строк; это делается так:

```
@tex
\global\emergencystretch = .9\hsize
@end tex
```

(Вы можете подобрать дробь по необходимости.) Такое огромное значение для `\emergencystretch` не может приниматься по умолчанию, потому что тогда набранный вывод был бы в основном заметно ниже качеством. Значение по умолчанию равно ‘`.15\hsize`’. `\hsize` — это размерность в TEX, содержащая текущую ширину строки.

Однако, для существующих переполненных боксов TEX будет печатать большой уродливый черный прямоугольник после строки, содержащей переполненный горизонтальный бокс, если не сказано иного. Это делается, чтобы вы заметили место, где возникла проблема, если вы корректируете черновик.

Для предотвращения таких ужасов в вашей окончательной распечатке, напишите следующее в начале Texinfo-файла на отдельной строке, перед командой `@titlepage`:

```
@finalout
```

## 19.11 Печать “маленьких” книг

По умолчанию, TEX набирает страницы для печати в формате 8.5 на 11 дюймов. Однако, вы можете указать TEX набирать документ в формате 7 на 9.25 дюймов, ко-

торый подходит для переплетенных книг, вставив следующую команду на отдельной строке в начале Texinfo-файла, перед титульным листом:

```
@smallbook
```

(Так как многие книги имеют размер примерно 7 на 9.25 дюймов, эту команду лучше было бы назвать `@regularbooksize`, но она стала называться командой `@smallbook` в сравнении с форматом 8.5 на 11 дюймов.)

Если вы пишите команду `@smallbook` между строк start-of-header и end-of-header, то в режиме Texinfo команда форматирования области с помощью TEX, `texinfo-tex-region`, будет форматировать область с размером "маленькой" книги (см. Раздел 3.2.2 [Начало заголовка], с. 30).

См. Раздел 10.6 [small], с. 89, для информации о командах, облегчающих создание примеров для меньших руководств.

См. Раздел 19.3 [Форматирование с texi2dvi], с. 148, и Раздел 19.9 [Подготовка к применению TEX], с. 152, другие способы отформатировать в формате `@smallbook`, не требующие изменения исходного файла.

## 19.12 Печать на формате A4

Вы можете сказать TEX форматировать документ для печати на бумаге европейского формата A4 с помощью команды `@afourpaper`. Пишите эту команду на отдельной строке недалеко от начала Texinfo-файла, до титульного листа. Например, так вы могли бы написать заголовок для данного руководства:

```
\input texinfo    @c -*-texinfo-*-
@c %**start of header
@setfilename texinfo
@settitle Texinfo
@afourpaper
@c %**end of header
```

См. Раздел 19.3 [Форматирование с texi2dvi], с. 148, Раздел 19.9 [Подготовка к применению TEX], с. 152, другие способы отформатировать в формате `@afourpaper`, не требующие изменения исходного файла.

Вы можете предпочесть или не предпочесть форматирование, получающееся с помощью команды `@afourlatex`. Есть также команда `@afourwide` для бумаги A4 в широком формате.

## 19.13 `@pagesizes` [*ширина*][, *высота*]: Произвольный размер страниц

Вы можете явно задать высоту и (возможно) ширину области основного текста на странице с помощью команды `@pagesizes`. Пишите ее на отдельной строке недалеко от начала Texinfo-файла, до титульного листа. Сначала пишется высота, потом, если нужно, ширина, разделенные запятыми. Примеры:

```
@pagesizes 200mm,150mm
```

и

```
@pagesizes 11.5in
```

Это может быть полезно при печати на формате B5. Подчеркнем, эта команда задает размер *области текста*, а не размер бумаги (который равен 250 mm на 177 mm для B5, 14 in на 8.5 in для legal).

Чтобы сделать более изощренные изменения, например изменение полей страницы, вы должны определить новую команду в ‘`texinfo.tex`’ (или ‘`texinfo.cnf`’, см. Раздел 19.9 [Подготовка к применению TEX], с. 152).

См. Раздел 19.3 [Форматирование с texi2dvi], с. 148, and Раздел 19.9 [Подготовка к применению TEX], с. 152, другие способы задать команду `@pagesizes`, не требующие изменения исходного файла.

`@pagesizes` игнорируется `makeinfo`.

## 19.14 Обрезные метки и увеличение

С помощью команды `@cropmarks` вы можете (попытаться) заставить TEX печатать метки обреза листа в углах страниц. Пишите эту команду на отдельной строке между `@iftex` и `@end iftex` недалеко от начала Texinfo-файла, до титульного листа, как показано здесь:

```
@iftex
@cropmarks
@end iftex
```

Это команда преимущественно для принтеров, которые набирают несколько страниц на одном листе или одной пленке; но вы можете попытаться применить ее для пометки углов книги с размером 7 на 9.25 дюймов, установленным командой `@smallbook`. (Принтеры могут не напечатать обрезные метки для вывода обычного размера, напечатанного на обычного размера бумаге.) Так как различные печатающие устройства работают по-разному, вам придется с духом приключения поисследовать использование этой команды. Вам, возможно, понадобится переопределить эту команду в файле ‘`texinfo.tex`’.

Вы можете попытаться указать TEX печатать страницы большими или меньшими, чем обычно, с помощью команды TEX `\mag`. Весь набор масштабируется пропорционально больше или пропорционально меньше. (`\mag` обозначает “magnification”, то есть “увеличение”.) Это *не* `@`-команда Texinfo, а команда plain TEX, перед которой ставится обратная косая черта. Вы должны писать эту команду между `@tex` и `@end tex` (см. Раздел 16.3 [Команды прямого форматирования], с. 135).

После команды `\mag` пишите ‘`=`’ и затем число, равное желаемому увеличению, умноженному на 1000. Например, чтобы напечатать страницы размером в 1.2 от нормального, напишите следующее недалеко от начала Texinfo-файла, до титульного листа:

```
@tex
\mag=1200
@end tex
```

В некоторых технологиях печати вы можете напечатать копии нормального размера, выглядящие лучше, чем обычно, передав в типографию увеличенный оригинал-макет. Они производят уменьшение, улучшая в действительности разрешение.

В зависимости от вашей системы, DVI-файлы, подготовленные с помощью нестандартной `\mag`, могут не печататься или печататься только при определенных увеличениях. Будьте готовы к экспериментам.

## 19.15 Вывод в PDF

Вы можете сгенерировать из исходного Texinfo-файла выходной PDF-файл, используя для обработки вашего файла программу `pdftex`, а не простой `tex`. Просто запустите '`pdftex foo.texi`' вместо '`tex foo.texi`' или задайте для `texi2dvi` ключ '`-pdf`' .

PDF означает Portable Document Format[2], он был изобретен фирмой Adobe Systems. Определение формата файлов доступно свободно, также доступен свободная программа просмотра для системы X Windows. Поскольку PDF — это двоичный формат, команды '`@ifpdf`' или '`@pdf`', по аналогии с другими выходными форматами, не существует.

Несмотря на слово 'переносимый' в названии, PDF-файлы близко не подходят по переносимости к форматам простого ASCII (Info, HTML), которые также поддерживаются Texinfo (о переносимости относительно DVI можно поспорить). Они также бывают намного больше, и в ни нет хорошей поддержки растровых шрифтов, используемых в TEX (по умолчанию). Тем не менее, PDF-файл показывает на экране действительный печатный документ как можно более правдиво, в отличие, скажем, от HTML, так что оба нужны.

Поддержа PDF в Texinfo довольно рудиментарна.

[2] Переносимый формат документов. *(Прим. переводчика)*

# 20 Создание и установка Info-файлов

Эта глава рассказывает, как создать и установить Info-файлы. См. Раздел 1.3 [Info-файлы], с. 5, для получения общих сведений о самом формате этих файлов.

## 20.1 Создание Info-файла

`makeinfo` — это программа, преобразующая Texinfo-файл в Info-файл, файл на HTML, или простой текст.

`texinfo-format-region` и `texinfo-format-buffer` — это функции GNU Emacs, преобразующие Texinfo в Info.

Информацию об установке Info-файла в систему Info смотрите в см. Раздел 20.2 [Установка Info-файла], с. 167.

### 20.1.1 Преимущества `makeinfo`

Утилита `makeinfo` создает Info-файл из исходного Texinfo-файла быстрее любой форматирующей команды Emacs и предоставляет лучшие сообщения об ошибках. Мы рекомендуем применять ее. `makeinfo` — это программа на Си, независимая от Emacs. Вы не должны запускать Emacs, чтобы использовать `makeinfo`, что означает, что вы можете использовать `makeinfo` на машинах, слишком маленьких, чтобы запустить Emacs. Вы можете запустить `makeinfo` любым из трех таких способов: из оболочки операционной системы, из оболочки внутри Emacs или напечатав команды *`C-c C-m C-r`* или *`C-c C-m C-b`* в режиме Texinfo в Emacs.

Команды `texinfo-format-region` и `texinfo-format-buffer` полезны, если вы не можете запустить `makeinfo`. Кроме того, при некоторых обстоятельствах они форматируют небольшие области или буферы быстрее, чем `makeinfo`.

### 20.1.2 Запуск `makeinfo` из оболочки

Чтобы создать Texinfo-файл из Info-файла, напечатайте `makeinfo` и имя Texinfo-файла. Таким образом, чтобы создать Info-файл для Bison, напечатайте в оболочке следующее:

```
makeinfo bison.texinfo
```

(Вы можете запустить оболочку в Emacs, напечатав *`M-x shell`*.)

### 20.1.3 Ключи для `makeinfo`

Команда `makeinfo` принимает несколько ключей. Чаще всего ключи применяются для задания колонки заполнения и стиля сносок. Каждый ключ командной строки — это слово с предшествующими символами ‘`-`’ или буква с предшествующим ‘`-`’. Вы можете использовать сокращения для длинных ключей, если они однозначны.

Например, вы можете использовать следующую команду для создания из файла ‘`bison.texinfo`’ такого Info-файла, в котором каждая строка заполнена на 68 колонок:

```
makeinfo -fill-column=68 bison.texinfo
```

Вы можете последовательно написать два ключа или более, например так:

```
makeinfo -no-split -fill-column=70 ...
```

Это сохранит Info-файл целиком в виде, возможно, очень большого файла и установит колонку заполнения равной 70.

Воспринимаются такие ключи:

‘`-D` *переменная*’
: Делает *переменную* определенной. Это эквивалентно `@set` *переменная* в Texinfo-файле (см. Раздел 16.4 [set clear value], с. 136).

‘`-commands-in-node-names`’
: Разрешает использование `@`-команд в именах нод. Это не рекомендуется, поскольку, вероятно, никогда не будет реализовано в TEX. Это также сильно замедляет работу `makeinfo`. Кроме того, этот ключ игнорируется, если использован ‘`-no-validate`’. См. Раздел 20.1.4 [Проверка указателей], с. 162, для дальнейших подробностей.

‘`-error-limit=`*предел*’
‘`-e` *предел*’
: Устанавливает наибольшее число ошибок, о которых сообщит `makeinfo` перед выходом (в предположении, что продолжать бесполезно); по умолчанию 100.

‘`-fill-column=`*ширина*’
‘`-f` *ширина*’
: Устанавливает наибольшее число колонок в строке; это правый край строки. Заполнение абзацев происходит по этой ширине. (Заполнение — это процесс разбиения и слияния строк таким образом, чтобы они имели длину, меньшую или равную числу, заданному в качестве колонки заполнения. Строки разбиваются между словами.) Значение по умолчанию равно 72. Игнорируется с ‘`-html`’.

‘`-footnote-style=`*стиль*’
‘`-s` *стиль*’
: Устанавливает стиль сносок равным *стилю*, ‘`end`’ для сносок в конце ноды (по умолчанию) или ‘`separate`’ для отдельных сносок. Значение, заданное этим ключом, перекрывает установленное в Texinfo-файле командой `@footnotestyle` (см. Раздел 13.10 [Сноски], с. 113). Когда стиль сносок установлен в значение ‘`separate`’, `makeinfo` создает новую ноду для сносок, находящихся в текущей ноде. Когда стиль сносок установлен в значение ‘`end`’, `makeinfo` помещает сноски в конце текущей ноды. Игнорируется ‘`-html`’.

‘`-force`’
‘`-F`’
: Обычно, если входной файл содержит ошибки, выходные файлы не создаются. С этим ключом, они сохраняются.

‘`-help`’
‘`-h`’
: Напечатать сообщение об использовании, перечисляющее все доступные ключи, и завершить выполнение успешно.

‘`-html`’
: Создавать вывод в формате HTML, а не Info. См. Раздел 20.1.9 [makeinfo html], с. 166.

‘`-I` *каталог*’
: Добавить *каталог* в конец списка каталогов для поиска файлов, включаемых командой `@include`. По умолчанию, `makeinfo` производит поиск только в текущем каталоге. Если *каталог* не задан, добавляется каталог ‘`.`’, то есть текущий. Заметьте, что *каталог* может на самом деле быть списком нескольких каталогов, разделенных обычным символом-разделителем путей (‘`:`’ в Unix, ‘`;`’ в MS-DOS/MS-Windows).

‘`-macro-expand=`*файл*’
‘`-E` *файл*’
: Вывести исходный Texinfo-файл, раскрыв все макросы, в заданный файл. Обычно результаты раскрытия макросов используются внутри `makeinfo` и затем отбрасываются. Этот ключ используется программой `texi2dvi`, если у вас установлена старая версия ‘`texinfo.tex`’, которая не поддерживет `@macro`.

‘`-no-headers`’
: Для вывода в формате Info, не включать меню и строки нод и писать на стандартный вывод (если не задан ключ ‘`-output`’). Это дает в результате ASCII-файл, который вы не можете читать в Info, так как он не содержит необходимых нод и меню. Прежде всего, он нужен, чтобы извлечь определенные фрагменты руководства в отдельные файлы, включаемые в пакет, такие, как файлы ‘`INSTALL`’.

  При выводе в формате HTML, если также задан ‘`-no-split`’, не включать ссылки для навигации в начало каждой ноды. См. Раздел 20.1.9 [makeinfo html], с. 166.

‘`-no-split`’
: Пропустить стадию разбиения в работе `makeinfo`. По умолчанию, большие выходные файлы (размером более 70 тысяч байт) разбиваются на меньшие подфайлы. При выводе в формате Info, размер каждого примерно 50 тысяч байт. При выводе в формате HTML, каждый файл содержит одну ноду (см. Раздел 20.1.9 [makeinfo html], с. 166).

‘`-no-pointer-validate`’
‘`-no-validate`’
: Пропустить стадию проверки указателей в работе `makeinfo`. Это можно сделать также командой `@novalidate` (см. Раздел 19.1 [Используйте TEX], с. 146). Обычно после обработки Texinfo-файла производятся некоторые проверки соответствия, чтобы убедиться, что перекрестные ссылки разрешаются и прочее. См. Раздел 20.1.4 [Проверка указателей], с. 162.

‘`-no-warn`’
: Подавить вывод предупреждений (но *не* сообщений об ошибках). Вы можете захотеть этого, если созданный вами файл содержит примеры перекрестных ссылок Texinfo, и ссылаемые ноды на самом деле не существуют.

‘`-number-sections`’
: Выводить номера глав, разделов и приложений как в печатном руководстве.

‘`-no-number-footnotes`’
: Подавить автоматическую нумерацию сносок. По умолчанию `makeinfo` последовательно нумерует все сноски в одной ноде, сбрасывая номер сноски на 1 в начале каждой ноды.

‘`-output=`*файл*’
‘`-o` *файл*’
: Указывает, что вывод должен быть направлен в *файл*, а не в файл, заданный командой `@setfilename` в исходном Texinfo-файле (см. Раздел 3.2.3 [setfilename], с. 30). Если *файл* равен ‘`-`’, вывод идет в стандартный вывод и подразумевается ‘`-no-split`’. При выводе в формате HTML, *файл* — это имя выходного файла для первой ноды (см. Раздел 20.1.9 [makeinfo html], с. 166).

‘`-P` *каталог*’
: Добавить *каталог* в начало списка каталогов для поиска файлов, включаемых командой `@include`. Если *каталог* не задан, добавляется каталог ‘`.`’, то есть текущий. Смотрите ‘`-I`’ для получения деталей.

‘`-paragraph-indent=`*отступ*’
‘`-p` *отступ*’
: Устанавливает стиль отступов в абзацах равным *отступу*. Значение, заданное этим ключом, перекрывает установленное в Texinfo-файле командой `@paragraphindent` (см. Раздел 3.2.6 [paragraphindent], с. 33). Значение *отступа* интерпретируется следующим образом:

  ‘`asis`’
  : Сохранять существующие отступы в начале абзацев.

  ‘`0`’ или ‘`none`’
  : Удалять существующие отступы.

  *число*
  : Делать в каждом абзаце отступы, равные заданному числу пробелов.

‘`-reference-limit=`*предел*’
‘`-r` *предел*’
: Устанавливает число ссылок на ноду, которое `makeinfo` будет делать, не выдавая предупреждения. Если нода содержит больше ссылок, чем это число, `makeinfo` создаст эти ссылки, но также выдаст предупреждение. Значение по умолчанию равно 1000.

‘`-U` *переменная*’
: Делает *переменную* неопределенной. Это эквивалентно `@clear` *переменная* в Texinfo-файле (см. Раздел 16.4 [set clear value], с. 136).

‘`-verbose`’
: Заставляет `makeinfo` выводить сообщения о том, что она делает. Обычно `makeinfo` выводит сообщения, только если есть ошибки или предупреждения.

‘`-version`’
‘`-V`’
: Напечатать номер версии и завершить выполнение успешно.

### 20.1.4 Проверка указателей

Если вы не подавили проверку указателей с помощью ключа '`-no-validate`' или команды `@novalidate` в исходном файле (см. Раздел 19.1 [Используйте TEX], с. 146), `makeinfo` сделает проверку конечного Info-файла. Как правило это означает проверку того, что все ноды, на которые вы ссылались, на самом деле существуют. Вот полный список проверок:

1. Если указатели на ноды 'Next', 'Previous' или 'Up' являются ссылками на ноды в текущем файле, но не внешними ссылками, такими, как на '`(dir)`', то ссылаемая нода должна существовать.
2. В каждой ноде, если нода 'Previous' отлична от ноды 'Up', то нода, на которую указывает поле 'Previous', должна содержать в поле 'Next' указатель обратно на эту ноду.
3. Каждая нода, за исключением 'Top', должна иметь указатель 'Up'.
4. Нода, на которую ссылается указатель 'Up' должна сама ссылаться на текущую ноду через пункт меню, за исключением случаев, когда нода, на которую ссылается 'Up' имеет форму '(*файл*)'.
5. Если в ноде указатель 'Next' не тот же, что и указатель 'Next' ноды 'Up', то нода, на которую ссылается 'Next' должна иметь указатель 'Previous', ссылающийся обратно на текущую ноду. Это правило позволяет последней ноде в разделе ссылаться на на первую ноду следующей главы.
6. На каждую ноду, за исключением 'Top', должна быть ссылка по крайней мере из одной другой ноды, или через указатели 'Previous' или 'Next', или через меню или перекрестную ссылку.

Некоторые документы Texinfo могут не пройти фазу проверки, потому что в них команды вроде `@value` и `@definfoenclose` использовались в определениях нод и в перекрестных ссылках непоследовательно. Рассмотрим следующий пример:

```
@set nodename Node 1

@node @value{nodename}, Node 2, Top, Top

Это нода 1.

@node Node 2, , Node 1, Top

Это нода 2.
```

Эдесь на ноду "Node 1" ссылаются как по точному имени, так и через `@value`.

По умолчанию `makeinfo` терпит в таких случаях неуспех, поскольку имена нод не раскрываются полностью, пока не будут записаны в выходной файл. Вы должны всегда стараться ссылаться на ноды последовательно; скажем, в примере выше во второй строка `@node` следует также использовать `@value`. Если однако, по какой-либо причине вы *должны* ссылаться на имена нод непоследовательно, и `makeinfo` не подтверждает правильность этого файла, вы можете использовать ключ '`-commands-in-node-names`', чтобы принудить `makeinfo` производить ресурсоемкое раскрытие всех имен нод, которые она находит в документе. Это, однако, может заметно замедлить работу програм-

мы; для больших файлов, таких как the Jargon file, было зафиксировано двукратное увеличение времени преобразования.

Поддержка `@`-команд в директивах `@node` не обладает достаточной общностью, чтобы ей можно было свободно пользоваться. Например, если приведенный выше пример переопределит бы где-то в документе `nodename`, то `makeinfo` не сможет преобразовать ее, даже если вызвать с ключом '`-commands-in-node-names`'.

'`-commands-in-node-names`' не имеет действия, если задан ключ '`-no-validate`'.

### 20.1.5 Запуск `makeinfo` из Emacs

Вы можете запустить `makeinfo` из режима Texinfo в GNU Emacs, используя команду `makeinfo-region` или `makeinfo-buffer`. В режиме Texinfo, эти команды по умолчанию привязаны к *C-c C-m C-r* и *C-c C-m C-b*.

*C-c C-m C-r*
*M-x makeinfo-region*
: Форматировать текущую область для Info.

*C-c C-m C-b*
*M-x makeinfo-buffer*
: Форматировать текущий буфер для Info.

Когда вы вызываете `makeinfo-region` или `makeinfo-buffer`, Emacs запрашивает имя файла, предлагая по умолчанию имя текущего файла. Вы можете, если хотите, отредактировать в минибуфере имя файла по умолчанию, перед нажатием ⟨RET⟩ и началом процесса `makeinfo`.

Команды Emacs `makeinfo-region` и `makeinfo-buffer` запускают программу `makeinfo` во временном буфере оболочки. Если `makeinfo` находит ошибки, Emacs отображает сообщения об ошибках в этом временном буфере.

Вы можете произвести грамматический разбор сообщений об ошибках, напечатав *C-x `* (`next-error`). Это заставляет Emacs перевести курсор на ту строку в исходном Texinfo-файле, которая, по мнению `makeinfo` вызывает ошибку. См. раздел "Running `make` or Compilers Generally" в *Руководство по GNU Emacs*, для подробной информации об использовании команды `next-error`.

Помимо этого, вы можете уничтожить оболочку, в которой запущена команда `makeinfo`, или сделать так, чтобы буфер оболочки показывал самые последние строки вывода.

*C-c C-m C-k*
*M-x makeinfo-kill-job*
: Уничтожить запущенный в данный момент процесс `makeinfo`, созданный `makeinfo-region` или `makeinfo-buffer`.

*C-c C-m C-l*
*M-x makeinfo-recenter-output-buffer*
: Показать в буфере оболочки `makeinfo` самые последние строки вывода.

(Обратите внимание, аналогичные команды для уничтожения и центрирования процесса TEX — это *C-c C-t C-k* и *C-c C-t C-l*. См. Раздел 19.6 [Печать в режиме Texinfo], с. 150.)

Вы можете задать ключи для `makeinfo`, установив переменную `makeinfo-options` с помощью команд *`M-x edit-options`* или *`M-x set-variable`*, или установив эту переменную в вашем файле инициализации '`.emacs`'.

Например, вы можете написать в вашем файле '`.emacs`' следующее:

```
(setq makeinfo-options
      "-paragraph-indent=0 -no-split
       -fill-column=70 -verbose")
```

Дальнейшую информацию смотрите в Раздел 20.1.3 [Ключи `makeinfo`], с. 158, а также "Editing Variable Values," "Examining and Setting Variables," и "Init File" в *Руководстве по GNU Emacs*.

### 20.1.6 Команды `texinfo-format...`

В режиме Texinfo GNU Emacs вы можете отформатировать Texinfo-файл или его часть с помощью команды `texinfo-format-region`. Она форматирует текущую область и выводит форматированный текст во временный буфер, называемый '`*Info Region*`'.

Аналогично, вы можете отформатировать буфер с помощью команды `texinfo-format-buffer`. Она делает новый буфер и выводит в него Info-файл. Вызов *`C-x C-s`* сохранит этот Info-файл под именем, заданным строкой `@setfilename`, которая должна быть недалеко от начала Texinfo-файла.

*`C-c C-e C-r`*
`texinfo-format-region`
: Форматировать текущую область для Info.

*`C-c C-e C-b`*
`texinfo-format-buffer`
: Форматировать текущий буфер для Info.

Команды `texinfo-format-region` и `texinfo-format-buffer` обеспечивают некоторую проверку ошибок, а другие функции могут предоставить вам дальнейшую помощь в нахождении ошибок форматирования. Эти процедуры описаны в приложении; смотрите Приложение G [Поиск ошибок], с. 211. Однако программа `makeinfo` часто работает быстрее и предоставляет лучшую проверку ошибок (см. Раздел 20.1.5 [makeinfo в Emacs], с. 163).

### 20.1.7 Пакетное форматирование

Вы можете отформатировать Texinfo-файлы для Info, используя `batch-texinfo-format` и пакетный режим Emacs. Вы можете запустить Emacs в пакетном режиме из оболочки, в том числе из оболочки внутри Emacs. (См. раздел "Command Line Switches and Arguments" в *Руководство по GNU Emacs*.)

Вот команда оболочки для форматирования всех файлов в текущем каталоге, оканчивающихся на '`.texinfo`':

```
emacs -batch -funcall batch-texinfo-format *.texinfo
```

Emacs обрабатывает все файлы, перечисленные в командной строке, даже если во время форматирования некоторых возникли ошибки.

Запускайте `batch-texinfo-format` только в пакетном режиме Emacs, как показано; эта команда не интерактивна. Она уничтожает Emacs после завершения.

`batch-texinfo-format` удобна, если у вас нет `makeinfo`, и вы хотите отформатировать несколько Texinfo-файлов одновременно. Когда вы используете пакетный режим, вы создаете новый процесс Emacs. Это высвобождает ваш текущий Emacs, так что вы можете продолжать в нем работать. (Когда вы запускаете `texinfo-format-region` или `texinfo-format-buffer`, вы не можете использовать Emacs ни для чего больше, пока команда не закончит работу.)

### 20.1.8 Создание тегов и разбиение файлов

Если Texinfo-файл содержит более 30000 байт, `texinfo-format-buffer` автоматически создает таблицу тегов для его Info-файла; `makeinfo` всегда создает таблицу тегов. С *таблицей тегов* Info может переходить к другим нодам быстрее, чем без нее.

Кроме того, если Texinfo-файл содержит более примерно 70000 байт, `texinfo-format-buffer` и `makeinfo` разбивают большой Info-файл на меньшие *косвенные* подфайлы примерно по 50000 байт в каждом. Большие файлы разбиваются на меньшие, чтобы Emacs не должен был делать большие буферы для хранения большого Info-файла целиком; вместо этого Emacs выделяет памяти чтобы как раз хватило для маленького, отсоединенного файла, нужного в данный момент. Таким способом Emacs избегает ненужной траты памяти, когда вы запускаете Info. (До того, как было реализовано разбиение, Info-файлы всегда делались короткими, а для создания единого, большого печатного руководства из меньших Info-файлов были разработаны *включаемые файлы*. См. Приложение E [Включаемые файлы], с. 202, для подробной информации. Включаемые файлы до сих пор используются для очень больших документов, таких как *The Emacs Lisp Reference Manual*, в котором каждая глава — это отдельный файл.)

Когда файл разбит, сама Info использует укороченную версию первоначального файла, содержащую только таблицу тегов и ссылки на отсоединенные файлы. Отсоединенные файлы называются *косвенными* файлами.

Отсоединенные файлы имеют имена, образованные добавлением ‘`-1`’, ‘`-2`’, ‘`-3`’ и так далее к имени файла, заданного командой `@setfilename`. Укороченная версия первоначального файла так же имеет имя, заданное `@setfilename`.

На одной стадии написания оригинала этого документа, к примеру, Info-файл был сохранен как ‘`test-texinfo`’, и этот файл выглядел следующим образом:

```
Info file: test-texinfo,    -*-Text-*-
produced by texinfo-format-buffer
from file: new-texinfo-manual.texinfo

^_
Indirect:
test-texinfo-1: 102
test-texinfo-2: 50422
```

```
test-texinfo-3: 101300
^_^L
Tag table:
(Indirect)
Node: overview^?104
Node: info file^?1271
Node: printed manual^?4853
Node: conventions^?6855
...
```

(Но в ‘`test-texinfo`’ было гораздо больше нод, чем показано здесь.) Каждый из отсоединенных, косвенных файлов, ‘`test-texinfo-1`’, ‘`test-texinfo-2`’ и ‘`test-texinfo-3`’, перечислен в этом файле после строки, в которой написано ‘`Indirect:`’. Таблица тегов помещена после строки ‘`Tag table:`’.

В перечне косвенных файлов, число, следующее после имени файла, хранит накопленное количество байт в предыдущих косвенных файлах, не считая сам перечень файлов, таблицу тегов или текст разрешения на копирование в каждом файле. В таблице тегов, число, следующее после имени ноды, хранит позицию начала этой ноды, в байтах от начала (разбиваемого) файла.

Если вы используете для создания Info-файлов `texinfo-format-buffer`, вы, возможно, захотите запустить команду `Info-validate`. (Команда `makeinfo` сама по себе делает хорошие проверки, вам не нужно применять `Info-validate`.) Однако, вы не можете запустить команду проверки нод *`M-x Info-validate`* в косвенных файлах. Для получения информации о том, как предотвратить разбиение файла и как проверить структуру нод, смотрите Раздел G.5.1 [Использование Info-validate], с. 216.

### 20.1.9 Создание HTML

Как альтернативу обычному выводу в формате Info, вы можете использовать ключ ‘`-html`’ для создания вывода в формате HTML для установки на Web-сайте (к примеру). В данном выпуске `makeinfo` производит монолитный HTML-файл; разбиение по главам или нодам не поддерживается. Мы надеемся в скором времени реализовать эту возможность.

Выходному HTML-файлу дается имя в соответствии с `@setfilename`, но расширение ‘`.info`’ заменяется на ‘`.html`’.

Входной текст Texinfo, помеченный командой `@ifhtml`, будет давать вывод, только если задан ключ ‘`-html`’. Входной текст, помеченный `@html`, передается на выход буквально (с подавлением обычного преобразования входных символов ‘`<`’, ‘`>`’ и ‘`&`’, имеющих особое значение в HTML).

Ключ ‘`-footnote-style`’ на данный момент игнорируется при выводе HTML; для сносок делаются гиперсвязи на конец выходного файла.

Создаваемый HTML главным образом соответствует стандарту (то есть HTML 2.0, RFC1866). Исключение в том, что из команды `@multitable` создаются таблицы HTML 3.2, но так, чтобы они как можно лучше отображались броузерами без поддержки таблиц. Пожалуйста, сообщайте о выводе, полученном от `makeinfo` без сообщений об ошибок, но нарушающем HTML 3.2 DTD как об ошибке программы.

В начало нод, как при выводе в Info, вставляются панели навигации. Это можно предотвратить, используя ключ ‘`-no-headers`’ вместе с ‘`-no-split`’. Элементы заголовка `<link>` могут поддерживать перемещение, как в Info, с броузерами, реализующими эту возможность HTML 1.0, такими как Lynx и Emacs W3. Все же вы обычно теряете возможность поиска регулярных выражений и вхождений именного указателя во многих файлах, предоставляемую программами чтения Info. Вместо этого, когда возможно, из команд Texinfo создаются гиперсвязи. Команды ‘`@xref`’, ссылающиеся на другие документы, создаются в предположении, что они также доступны в форме HTML, и к имени Info-файла в ‘`@xref`’ добавляется ‘`.html`’. Предположительно, часто это не будет работать.

## 20.2 Установка Info-файла

Info-файлы обычно хранятся в каталоге ‘`info`’. Вы можете читать Info-файлы с помощью отдельной программы Info или программой чтения, встроенной в Emacs. (См. Info файл ‘`info`’, node ‘`Top`’, введение в Info.)

### 20.2.1 Файл-каталог ‘`dir`’

Чтобы Info могла работать, в каталоге ‘`info`’ должен быть файл, служащий каталогом верхнего уровня системы Info. По соглашению, этот файл называется ‘`dir`’. (Вы можете узнать местоположение этого файла, нажав в Emacs *`C-h i`*, чтобы войти в Info, и затем *`C-x C-f`*, чтобы увидеть полный путь к каталогу ‘`info`’.)

Файл ‘`dir`’ сам по себе является Info-файлом. В нем содержится меню верхнего уровня для всех Info-файлов в системе. Это меню выглядит следующим образом:

```
* Menu:

* Info:    (info).     Система просмотра документации.
* Emacs:   (emacs).    Расширяемый, самодокументированный
                       текстовый редактор.
* Texinfo: (texinfo).  Создание печатного руководства с
                       помощью TeX или Info-файла из
                       одного исходного файла.
...
```

Каждый из этих пунктов меню указывает на первую (‘Top’) ноду Info-файла, чье имя написано в круглых скобках. (В пункте меню не обязательно задавать ноду ‘Top’, так как если нода не задана, Info сама переходит к ноде ‘Top’. См. Раздел 7.5 [Ноды в других Info-файлах], с. 64.)

Таким образом, пункт ‘`Info`’ указывает на первую ноду файла ‘`info`’, а пункт ‘`Emacs`’ — на первую ноду файла ‘`emacs`’.

В каждом Info-файле, указатель ‘Вверх’ (‘Up’) первой ноды ссылается на файл `dir`. Например, строка первой ноды руководства по Emacs выглядит в Info так:

```
File: emacs  Node: Top, Up: (DIR), Next: Distrib
```

В данном случае имя файла `dir` написано заглавными буквами — оно может быть написано как заглавными, так и строчными. В общем случае это не так, но для ‘`dir`’ делается исключение.

### 20.2.2 Включение нового Info-файла

Чтобы включить новый Info-файл в систему, вы должны написать пункт меню и добавить его в файл ‘dir’ в каталоге ‘info’. Например, если вы добавляете документацию для GDB, вам следует написать такой новый пункт:

```
* GDB: (gdb).           Отладчик C на уровне исходного кода.
```

Первая часть пункта меню — это название этого пункта, за которым следует двоеточие. Во второй части записано имя Info-файла в круглых скобках, и за ним точка. Третья часть представляет собой описание.

Имя Info-файла обычно имеет расширение ‘.info’. Значит, Info-файл для GDB мог бы называться ‘gdb’ или ‘gdb.info’. Программы чтения Info автоматически пробуют имена файлов и с расширением ‘.info’[1], и без него; поэтому лучше избегать написания лишнего текста и не писать в пункте меню ‘.info’ явно. Например, в пункте меню GDB в качестве имени файла нужно использовать ‘gdb’, а не ‘gdb.info’.

### 20.2.3 Info-файлы в других каталогах

Если Info-файл не находится в каталоге ‘info’, есть три способа указать его местоположение:

1. Написать полный путь в файле ‘dir’ во второй части пункта меню.
2. Если вы пользуетесь Emacs, включить имя файла во второй файл ‘dir’ в его каталоге; и потом добавить имя этого каталога в переменную `Info-directory-list` в вашем собственном или локальном для вашей машины файле инициализации.

   Эта переменная сообщает Emacs где искать файлы ‘dir’ (эти файлы должны обязательно называться ‘dir’). Emacs объединяет файлы с именем ‘dir’ из каждого из перечисленных каталогов. (В Emacs версии 18, вы можете записать в переменную `Info-directory` имя только одного каталога.)
3. Указать имя каталога Info в переменной среды `INFOPATH` в вашем файле инициализации ‘.profile’ или ‘.cshrc’. (Info-файлы, чье местоположение задается таким способом, сможете найти только вы и те, кто так же установил эту переменную.)

Например, чтобы достичь тестового файла в каталоге ‘/home/bob/info/’, вы можете добавить подобный пункт в меню стандартного файла ‘dir’:

```
* Тест: (/home/bob/info/info-test).  Тестовый файл Боба.
```

В данном случае абсолютное имя файла ‘info-test’ написано в качестве второй части пункта меню.

Или вы можете написать такую команду в вашем файле ‘.emacs’:

```
(require 'info)
(setq Info-directory-list
      (cons (expand-file-name "/home/bob/info") Info-directory-list))
```

Она велит Emacs объединить файл ‘dir’ из каталога ‘/home/bob/info’ с системным файлом ‘dir’. Info перечислит файл ‘/home/bob/info/info-test’ как пункт меню файла ‘/home/bob/info/dir’. Emacs производит объединение, только когда *M-x info* запущена первый раз, так что если вы хотите установить `Info-directory-list` в

[1] В системах MS-DOS/MS-Windows, Info также попробует файл с расширением ‘.inf’.

сессии Emacs, где вы уже запускали `info`, вы должны выполнить `(setq Info-dir-contents nil)`, чтобы принудить Emacs заново составить файл '`dir`'.

И наконец, вы можете сообщить Info путь поиска, установив переменную среды `INFOPATH` в файле инициализации вашей оболочки, таком как '`.cshrc`', '`.profile`' или '`autoexec.bat`'. Если вы пользуетесь оболочкой, совместимой с Bourne shell, такой как `sh` или `bash`, вам нужно установить переменную среды `INFOPATH` в файле инициализации '`.profile`'; но если вы пользуетесь `csh` или `tcsh`, вы должны установить эту переменную в файле '`.cshrc`'. В системах MS-DOS/MS-Windows, вы должны установить `INFOPATH` в вашем файле '`autoexec.bat`' или в Реестре. Каждый тип командного интерпретатора используют различный синтаксис.

- В файле '`.cshrc`', вы можете установить переменную `INFOPATH` так:

```
setenv INFOPATH .:~/info:/usr/local/emacs/info
```

- В файле '`.profile`', вы можете получить тот же результат, написав:

```
INFOPATH=.:$HOME/info:/usr/local/emacs/info
export INFOPATH
```

- В файле '`autoexec.bat`' вы пишите такую команду[2]:

```
set INFOPATH=.;%HOME%/info;c:/usr/local/emacs/info
```

Как обычно, '`.`' обозначает текущий каталог. Emacs использует переменную среды `INFOPATH` для инициализации значения переменной Emacs `Info-directory-list`. Самостоятельная программа чтения Info сливает все файлы, называемые '`dir`', во всех каталогах, перечисленных в переменной среды `INFOPATH`, в одно меню, представляемое вам в ноде, называемой '`(dir)Top`'.

Какое бы значение вы не установили для `INFOPATH`, если его последним символом является двоеточие[3], оно замещается путем по умолчанию (задаваемым при компиляции). Это дает вам возможность дополнять путь по умолчанию, избавляя вас от необходимости перечислять все стандартные каталоги. Например (с использованием синтаксиса `sh`):

```
INFOPATH=/local/info:
export INFOPATH
```

приводит к поиску сначала в '`/local/info`', а потом в стандартных каталогах. Двойные двоеточия или двоеточия в начале не рассматриваются особо.

Когда вы создаете свой собственный файл '`dir`' для использования с `Info-directory-list` или `INFOPATH`, проще всего начать с копирования существующего файла '`dir`' и замены всего текста после '`* Menu:`' вашими желаемыми пунктами. При таком способе сохранится пунктуация и специальные символы CTRL-_, которые необходимы для Info.

### 20.2.4 Установка файлов-каталогов Info

При установке в вашу систему Info-файла, вы можете использовать программу `install-info` для обновления файла-каталога Info '`dir`'. Обычно `install-info` за-

[2] Обратите внимание на использование '`;`' в качестве разделителя каталогов и на другой синтаксис для получения значений переменных среды.

[3] В системах MS-DOS/MS-Windows, используйте вместо двоеточия точку с запятой.

пускается make-файлом пакета сразу после копирования Info-файла в подходящее для установки место.

Чтобы Info-файл мог работать с `install-info`, вы должны использовать команды `@dircategory` и `@direntry` в исходном Texinfo-файле. Используйте `@direntry`...`@end direntry` для задания пунктов меню, добавляемых в файл-каталог Info, и `@dircategory` для задания той части каталога Info, в который их следует поместить. Вот как эти команды используются в данном руководстве:

```
@dircategory Система документации Texinfo
@direntry
* Texinfo: (texinfo).           Формат документации GNU.
* install-info: (texinfo)Вызов install-info. ...
...
@end direntry
```

Это дает в Info-файле следующее:

```
INFO-DIR-SECTION Система документации Texinfo
START-INFO-DIR-ENTRY
* Texinfo: (texinfo).           Формат документации GNU.
* install-info: (texinfo)Вызов install-info. ...
...
END-INFO-DIR-ENTRY
```

Программа `install-info` видит эти строки в Info-файле и так узнает, что нужно делать.

Всегда пишите команды `@direntry` и `@dircategory` недалеко от начала Texinfo-файла, до первой команды `@node`. Если вы напишете их после, `install-info` не заметит их.

Если вы используете `@dircategory` в исходном Texinfo-файле более одного раза, каждое применение задает ‘текущую’ категорию; все последующие команды `@direntry` будут добавлять новый пункт к этой категории.

Вот несколько рекомендуемых категорий `@dircategory`: ‘GNU packages’, ‘GNU programming tools’, ‘GNU programming documentation’, ‘GNU Emacs Lisp’, ‘GNU libraries’, ‘Linux’, ‘TeX’, ‘Individual utilities’. Идея состоит в том, чтобы включать ноду ‘вызов’ для каждой устанавливаемой пакетом программы в категории ‘Individual utilities’, а вхождение для руководства целиком — в другой подходящей категории.

### 20.2.5 Вызов install-info

Программа `install-info` вставляет вхождения меню из Info-файла в файл ‘`dir`’, каталог верхнего уровня системы Info (объяснения, как работает файл ‘`dir`’, смотрите в предыдущих разделах). Чаще всего она запускается в процессе установки программного обеспечения или при составлении каталога всех руководств в системе. Обзор использования:

`install-info` [*ключ*]... [*info-файл* [*dir-файл*]]

Если не заданы *info-файл* или *dir-файл*, то должны быть заданы ключи (описанные ниже), определяющие их. Для них нет значений по умолчанию, определяемых на стадии компиляции, и стандартный ввод не используется. `install-info` за один запуск может считать только один info-файл и записать один dir-файл.

Если *dir-файл* (будучи задан) не существует, `install-info` создает его, если возможно (без вхождений).

Если какой-нибудь входной файл сжат с помощью `gzip` (см. раздел “Invoking gzip” в *Gzip*), `install-info` автоматически распаковывает его для чтения. А если сжат *dir-файл*, `install-info` также автоматически оставляет его сжатым после записи всех изменений. Если не существует сам *dir-файл*, `install-info` пытается открыть ‘*dir-file*`.gz`’.

Ключи:

`-delete` Удалить из *dir-файла* вхождения для *info-файла*. Имя файла во вхождениях в *dir-файле* должно совпадать с *info-файлом* (за исключением допустимого суффикса ‘`.info`’ в любом из них). Не вставлять новых вхождений.

`-dir-file=`*имя*
`-d` *имя* Задает имя файла-каталога Info. Это эквивалентно использованию аргумента *dir-файл*.

`-entry=`*текст*
`-e` *текст* Вставить *текст* в качестве вхождения в каталог Info; *текст* должен иметь вид пункта меню Info плюс одна или более дополнительных строк, начинающихся с пробельных символов. Если вы зададите более одного вхождения, все они будут добавлены. Если вы не зададите вхождений, они будут определены из самого Info-файла.

`-help`
`-h` Показать сообщение об использовании, перечисляющее основные способы использования и все доступные ключи, и завершить выполнение успешно.

`-info-file=`*файл*
`-i` *файл* Задает Info-файл, который нужно установить в каталог. Это эквивалентно использованию аргумента *info-файл*.

`-info-dir=`*кат*
`-D` *кат* Задает каталог с файлом ‘`dir`’. Эквивалентно ‘`-dir-file=`*dir*`/dir`’.

`-item=`*текст*
Синоним ‘`-entry=`*текст*’. Вхождение в каталог Info на самом деле является пунктом меню.

`-quiet` Подавить вывод предупреждений.

`-remove`
`-r` Синоним ‘`-delete`’.

`-section=`*разд*
`-s` *разд* Поместить вхождения этого файла в раздел *разд* каталога. Если вы зададите более одного раздела, все вхождения будут добавлены в каждый из разделов. Если вы не зададите разделов, они будут определены из самого info-файла.

`-version`
`-V` Показать информацию о версии и завершить выполнение успешно.

# Приложение A Список @-команд

Здесь приведен алфавитный список @-команд Texinfo. Квадратные скобки, [ ], обозначают необязательные аргументы; многоточие, '...', обозначает повторяемый текст.

@*пробельный символ*
: Символ @, за которым следует пробел, табуляция или новая строка, дают нормальный растяжимый междусловный пробел. См. Раздел 13.2.3 [Несколько пробелов], с. 107.

@!
: Создает восклицательный знак, на самом деле завершающий предложение (обычно после последней буквы в предложении, являющейся заглавной). См. Раздел 13.2.2 [Завершение предложения], с. 106.

@"
@'
: Создают умляут или акцент ударения, соответственно, над следующим символом, как в ö и ó. См. Раздел 13.3 [Вставка акцентов], с. 108.

@*
: Принудительно разрывает строку. Не завершайте абзац, использующий @* командой @refill. См. Раздел 14.1 [Разрыв строки], с. 117.

@,{*c*}
: Создает акцент седиль под *c*, как в ç. См. Раздел 13.3 [Вставка акцентов], с. 108.

@-
: Указывает возможный перенос. См. Раздел 14.2 [Переносы], с. 118.

@.
: Выводит точку, на самом деле завершающую предложение (обычно после последней буквы в предложении, являющейся заглавной). См. Раздел 13.2.2 [Завершение предложения], с. 106.

@:
: Указывает TEX, что непосредственно предшествующая точка, вопросительный или восклицательный знак или двоеточие не завершают предложение. Запрещает TEX вставлять дополнительный пропуск, как в конце предложения. Эта команда не влияет на вывод в Info-файл. См. Раздел 13.2.1 [Незавершение предложения], с. 106.

@=
: Создает акцент макрон (черточку) над следующим символом, как в ō. См. Раздел 13.3 [Вставка акцентов], с. 108.

@?
: Создает вопросительный знак, на самом деле завершающий предложение (обычно после последней буквы в предложении, являющейся заглавной). См. Раздел 13.2.2 [Завершение предложения], с. 106.

@@
: Обозначает символ "at", '@'. См. Раздел 13.1 [Вставка @ и фигурных скобок], с. 105.

@^
@`
: Создают сиркомфлекс (шапочку) или акцент грав, соответственно, над следующим символом, как в ô. См. Раздел 13.3 [Вставка акцентов], с. 108.

@{
: Обозначает левую фигурную скобку, '{'. См. Раздел 13.1 [Вставка @ и фигурных скобок], с. 105.

`@}` Обозначает правую фигурную скобку, '}'. См. Раздел 13.1 [Вставка @ и фигурных скобок], с. 105.

`@~` Создает акцент тильду над следующим символом, как в Ñ. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@AA{}`
`@aa{}` Создает заглавную или строчную скандинавские буквы A с кружком, соответственно: Å, å. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@acronym{`*аббревиатура*`}`
Пометить заданное слово как аббревиатуру, то есть сокращение, записанное полностью заглавными буквами, как 'NASA'. См. Раздел 9.1.12 [acronym], с. 83.

`@AE{}`
`@ae{}` Создает заглавную или строчную лигатуру AE, соответственно: Æ, æ. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@afourpaper`
`@afourpaper`
`@afourwide`
Изменяет размеры страницы для формата A4. См. Раздел 19.12 [Формат A4], с. 155.

`@alias` *новая=существующая*
Делает команду '`@`*новая*' псевдонимом существующей команды '`@`*существующая*'. См. Раздел 18.4 [alias], с. 143.

`@anchor{`*имя*`}`
Определяет *имя* как текущую позицию, для использования в качестве назначения перекрестной ссылки. См. Раздел 6.5 [@anchor], с. 59.

`@appendix` *название*
Начинает приложение. Его название появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой звездочек. См. Раздел 5.5 [Команды @unnumbered и @appendix], с. 48.

`@appendixsec` *название*
`@appendixsection` *название*
Начинает раздел внутри приложения. Название раздела появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой знаков равенства. `@appendixsection` — это более длинная форма записи команды `@appendixsec`. См. Раздел 5.8 [Команды для разделов], с. 49.

`@appendixsubsec` *название*
Начинает подраздел внутри приложения. Его название появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой дефисов. См. Раздел 5.10 [Команды для подразделов], с. 50.

`@appendixsubsubsec` *название*
: Начинает подподраздел внутри приложения. Его название появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой точек. См. Раздел 5.11 [Команды subsub], с. 50.

`@asis`
: Используется после `@table`, `@ftable` и `@vtable` для печати первой колонки таблицы без выделения (“как есть”). См. Раздел 11.3 [Создание двухколоночных таблиц], с. 96.

`@author` *автор*
: Набирает имя *автора* прижатым влево и подчеркивает его. См. Раздел 3.4.3 [Команды `@title` и `@author`], с. 37.

`@b{`*текст*`}`
: Печатает *текст* **жирным** шрифтом. Не влияет на Info. См. Раздел 9.2.3 [Шрифты], с. 85.

`@bullet{}`
: Создает большую черную точку или наиболее похожий на нее символ. См. Раздел 13.4.2 [`@bullet`], с. 109.

`@bye`
: Останавливает форматирование файла. Форматирующие команды не видят содержимое файла после команды `@bye`. См. Глава 4 [Завершение файла], с. 43.

`@c` *комментарий*
: Начинает комментарий в Texinfo. Остаток строки не появляется ни в Info-файле, ни в печатном руководстве. Синоним `@comment`. См. Раздел 1.7 [Комментарии], с. 9.

`@cartouche`
: Выделяет пример или цитату, окружая его рамкой с закругленными углами. Парная с командой `@end cartouche`. Не влияет на Info. См. Раздел 10.11 [Рисование рамок вокруг примеров], с. 91.)

`@center` *строка-текста*
: Центрирует строку текста, следующего за командой. См. Раздел 3.4.2 [`@center`], с. 36.

`@centerchap` *строка-текста*
: Как `@chapter`, но центрирует заголовок главы. См. Раздел 5.4 [`@chapter`], с. 48.

`@chapheading` *название*
: Печатает заголовок как для главы в тексте, но не в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой звездочек. См. Раздел 5.6 [`@majorheading` и `@chapheading`], с. 48.

`@chapter` *название*
: Начинает главу. Название главы появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой звездочек. См. Раздел 5.4 [`@chapter`], с. 48.

`@cindex` *вхождение*
: Добавляет *вхождение* в указатель понятий. См. Раздел 12.1 [Определение вхождений именных указателей], с. 100.

`@cite{`*ссылка*`}`
: Выделяет название книги или другую ссылку, для которой нет сопутствующего Info-файла. См. Раздел 9.1.11 [@cite], с. 83.

`@clear` *флаг*
: Сбрасывает *флаг*, запрещая командам форматирования Texinfo обрабатывать текст между последовательными парами команд `@ifset` *флаг* и `@end ifset` и предотвращая раскрытие `@value{`*флаг*`}` в значение, в которое установлен *флаг*. См. Раздел 16.4 [@set @clear @value], с. 136.

`@code{`*образец-кода*`}`
: Выделяет текст, являющийся выражением, синтаксически завершенной лексемой программы или именем программы. См. Раздел 9.1.1 [@code], с. 77.

`@command{`*имя-команды*`}`
: Обозначает имя команды, такое как `ls`. См. Раздел 9.1.8 [@command], с. 82.

`@comment` *комментарий*
: Начинает комментарий в Texinfo. Остаток строки не появляется ни в Info-файле, ни в печатном руководстве. Синоним `@c`. См. Раздел 1.7 [Комментарии], с. 9.

`@contents`
: Печатает полное содержание. Не влияет на Info, в котором вместо этого используются меню. См. Раздел 4.2 [Создание содержания], с. 44.

`@copyright{}`
: Создает символ охраны авторских прав. См. Раздел 13.5.2 [@copyright], с. 109.

`@defcodeindex` *имя-именного-указателя*
: Определяет новый именной указатель и команду для добавления к нему. Печатает вхождения шрифтом `@code`. См. Раздел 12.5 [Определение новых именных указателей], с. 104.

`@defcv` *категория класс имя*
`@defcvx` *категория класс имя*
: Форматирует описание переменной, связанной с классом в объектно-ориентированном программировании. Принимает три аргумента: категорию определяемой переменной, класс, к которой она принадлежит, и ее имя. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@deffn` *категория имя аргументы...*
`@deffnx` *категория имя аргументы...*
: Форматирует описание функции, интерактивной команды или похожего объекта, который может принимать аргументы. `@deffn` принимает в качестве аргументов категорию определяемого объекта, имя объекта и его аргументы, если таковые имеются. См. Глава 15 [Определения], с. 121.

`@defindex` *имя-именного-указателя*

Определяет новый именной указатель и команду для добавления к нему. Печатает вхождения романским шрифтом. См. Раздел 12.5 [Определение новых именных указателей], с. 104.

`@definfoenclose` *новая-команда, перед, после,*

Создает новую @-команду для Info, которая помечает текст, заключая его между строк, стоящих перед и после текста. См. Раздел 18.5 [definfoenclose], с. 144.

`@defivar` *класс имя-переменной-экземпляра*
`@defivarx` *класс имя-переменной-экземпляра*

Эта команда форматирует описание переменной экземпляра в объектно-ориентированном программировании. Эта команда эквивалентна '`@defcv {Переменная экземпляр} ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@defmac` *имя-макроса аргументы...*
`@defmacx` *имя-макроса аргументы...*

Форматирует описание макроса. Эта команда эквивалентна '`@deffn Макро ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@defmethod` *класс имя-метода аргументы...*
`@defmethodx` *класс имя-метода аргументы...*

Форматирует описание метода в объектно-ориентированном программировании. Эта команда эквивалентна '`@defop Метод ...`'. Принимает в качестве аргументов имя класса этого метода, имя этого метода и его аргументы, если таковые имеются. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@defop` *категория класс имя аргументы...*
`@defopx` *категория класс имя аргументы...*

Форматирует описание операции в объектно-ориентированном программировании. `@defop` принимает в качестве аргументов общее название категории операции, имя класса этой операции, ее имя и аргументы, если таковые имеются. См. Глава 15 [Определения], с. 121, а также Раздел 15.4.5 [Абстрактные типы], с. 129.

`@defopt` *имя-пользовательского-параметра*
`@defoptx` *имя-пользовательского-параметра*

Форматирует описание пользовательского параметра. Эта команда эквивалентна '`@defvr {Пользовательский параметр} ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@defspec` *имя-специальной-формы аргументы...*
`@defspecx` *имя-специальной-формы аргументы...*

Форматирует описание специальной формы. Эта команда эквивалентна '`@deffn {Специальная форма} ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@deftp` *категория имя-типа атрибуты...*
`@deftpx` *категория имя-типа атрибуты...*

Форматирует описание типа данных. `@deftp` принимает в качестве аргументов категорию, имя типа (слово вроде '`int`' или '`float`') и затем имена атрибутов объектов этого типа. См. Глава 15 [Определения], с. 121, а также Раздел 15.4.6 [Типы данных], с. 131.

`@deftypefn` *классификация тип-данных имя аргументы...*
`@deftypefnx` *классификация тип-данных имя аргументы...*

Форматирует описание функции или похожего объекта, который принимает аргументы и имеет тип. `@deftypefn` принимает в качестве аргументов классификацию определяемого объекта, его тип, имя и аргументы, если таковые имеются. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@deftypefun` *тип-данных имя-функции аргументы...*
`@deftypefunx` *тип-данных имя-функции аргументы...*

Форматирует описание функции в языках с контролем типов. Эта команда эквивалентна '`@deftypefn Функция ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@deftypeivar` *класс тип-данных имя-переменной*
`@deftypeivarx` *класс тип-данных имя-переменной*

Форматирует описание типизированной переменной экземпляра в объектно-ориентированном программировании См. Глава 15 [Определения], с. 121, а также Раздел 15.4.5 [Абстрактные типы], с. 129.

`@deftypemethod` *класс тип-данных имя-метода аргументы...*
`@deftypemethodx` *класс тип-данных имя-метода аргументы...*

Форматирует описание типизированного метода в объектно-ориентированном программировании. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@deftypeop` *категория класс тип-данных имя аргументы...*
`@deftypeopx` *категория класс тип-данных имя аргументы...*

Форматирует описание типизированной операции в объектно-ориентированном программировании. См. Глава 15 [Определения], с. 121, а также Раздел 15.4.5 [Абстрактные типы], с. 129.

`@deftypevar` *тип-данных имя-переменной*
`@deftypevarx` *тип-данных имя-переменной*

Форматирует описание переменной в языках с контролем типов. Эта команда эквивалентна '`@deftypevr Переменная ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@deftypevr` *классификация тип-данных имя*
`@deftypevrx` *классификация тип-данных имя*

Форматирует описание чего-то, похожего на переменную в языках с контролем типов — объекта, в который записывается значение. Принимает в качестве аргументов классификацию описываемого объекта, его тип и

имя. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@defun` *имя-функции аргументы...*
`@defunx` *имя-функции аргументы...*
Форматирует описание функции. Эта команда эквивалентна '`@deffn Функция ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@defvar` *имя-переменной*
`@defvarx` *имя-переменной*
Форматирует описание переменной. Эта команда эквивалентна '`@defvr Переменная ...`'. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@defvr` *категория имя*
`@defvrx` *категория имя*
Форматирует описание переменных любого вида. `@defvr` принимает в качестве аргументов категорию объекта и его имя. См. Глава 15 [Определения], с. 121, а также Раздел 15.3 [Подробно о командах для определений], с. 123.

`@detailmenu`
Уберегает `makeinfo` от остановки на детальном списке нод в главном меню. См. Раздел 3.5.2 [Части главного меню], с. 41.

`@dfn{`*термин*`}`
Выделяет вводимый или определяемый термин. См. Раздел 9.1.10 [`@dfn`], с. 82.

`@dircategory` *часть-dir*
Задает часть меню каталога Info, в которую должен быть помещено вхождение данного файла. См. Раздел 20.2.4 [Установка каталогов Info], с. 169.

`@direntry`
Начинает вхождение в каталога Info для данного файла. Парная с `@end direntry`. См. Раздел 20.2.4 [Установка каталогов Info], с. 169.

`@display` Начинает разновидность примера. Похожа на `@example` (делает в тексте отступ, не заполняет), но не выбирает новый шрифт. Парная с `@end display`. См. Раздел 10.7 [`@display`], с. 90.

`@dmn{`*размер*`}`
Форматирует единицы измерения, как в 12 pt. Заставляет TEX вставить тонкий пробел перед *размером*. Не влияет на Info. См. Раздел 13.2.4 [`@dmn`], с. 107.

`@documentencoding` *кодировка*
Декларирует входную *кодировку*. См. Раздел 17.2 [`@documentencoding`], с. 140.

`@documentlanguage` *CC*
: Декларирует язык документа как двухсимвольное сокращение ISO-639 *CC*. См. Раздел 17.1 [@documentlanguage], с. 140.

`@dotaccent{`*c*`}`
: Создает акцент-точку над символом *c*, как в ȯ. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@dots{}` Вставляет многоточие: '`...`'. См. Раздел 13.4.1 [@dots], с. 109.

`@email{`*адрес*`[, `*отображаемый-текст*`]}`
: Обозначает адрес электронной почты. См. Раздел 9.1.14 [@email], с. 83.

`@emph{`*текст*`}`
: Выделяет *текст*; текст отображается *курсивом* в печатном руководстве и окружается подчерками в Info. См. Раздел 9.2 [Логическое ударение], с. 84.

`@end` *среда*
: Завершает *среду*, как в '`@end example`'. См. Раздел 1.5 [@-команды], с. 7.

`@env{`*переменная-среды*`}`
: Обозначает имя переменной среды, такой как `PATH`. См. Раздел 9.1.6 [@env], с. 81.

`@enddots{}`
: Создает многоточие, завершающее предложение, такое как это.... См. Раздел 13.4.1 [@dots{}], с. 109.

`@enumerate [`*число-или-буква*`]`
: Начинает нумерованный перечень, использующий для каждого пункта команду `@item`. Возможно, начинает перечень с *числа-или-буквы*. Парная с `@end enumerate`. См. Раздел 11.2 [@enumerate], с. 95.

`@equiv{}` Указывает читателю на точную эквивалентность двух форм с помощью знака '≡'. См. Раздел 13.9.5 [Обозначение эквивалентности], с. 112.

`@error{}` Указывает читателю с помощью особого знака, что последующий текст является сообщением об ошибке: 'error '. См. Раздел 13.9.4 [Обозначение сообщений об ошибках], с. 112.

`@evenfooting [`*левая*`] @| [`*средняя*`] @| [`*правая*`]`
`@evenheading [`*левая*`] @| [`*средняя*`] @| [`*правая*`]`
: Задает нижние и соотв. верхние заголовки для четных (левых) страниц. Допустима только внутри блока `@iftex`. См. Раздел F.3 [Как создать свои заголовки], с. 208.

`@everyfooting [`*левая*`] @| [`*средняя*`] @| [`*правая*`]`
`@everyheading [`*левая*`] @| [`*средняя*`] @| [`*правая*`]`
: Задает нижние и соотв. верхние заголовки страниц. Не влияет на Info. См. Раздел F.3 [Как создать свои заголовки], с. 208.

`@example` Начинает пример. Делает в тексте отступ, не заполняет его и выбирает равноширинный шрифт. Парная с `@end example`. См. Раздел 10.3 [@example], с. 87.

`@exampleindent` *отступ*
: Делает отступ в блоках, подобных примерам, на заданное число пробелов (возможно 0). См. Раздел 3.2.7 [Отступ в абзацах], с. 33.

`@exclamdown{}`
: Создает перевернутый восклицательный знак. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@exdent` *строка-текста*
: Удаляет любой отступ, который может быть в строке. См. Раздел 10.9 [Отмена отступа в строке], с. 90.

`@expansion{}`
: Показывает читателю результат раскрытия макроса с помощью специального знака: '↦'. См. Раздел 13.9.2 [Обозначение раскрытия], с. 111.

`@file{`*имя-файла*`}`
: Выделяет имя файла, буфера, ноды или каталога. См. Раздел 9.1.7 [@file], с. 81.

`@finalout`
: Запрещает TEX печатать большие черные прямоугольники-предупреждения рядом со слишком широкими строками. См. Раздел 19.10 [Переполненные боксы], с. 154.

`@findex` *вхождение*
: Добавляет *вхождение* в указатель функций. См. Раздел 12.1 [Определение вхождений именных указателей], с. 100.

`@flushleft`
`@flushright`
: Прижимает каждую строку влево, но оставляет правый край неровным. Шрифт не меняется. Парная с `@end flushleft`. `@flushright` аналогична. См. Раздел 10.10 [@flushleft и @flushright], с. 91.

`@footnote{`*текст-сноски*`}`
: Вводит сноску. TEX печатает текст сноски внизу страницы; Info может форматировать в стиле 'в конце ноды' или 'в отдельнойноде'. См. Раздел 13.10 [Сноски], с. 113.

`@footnotestyle` *стиль*
: Задает стиль сносок в Info-файле, либо '`end`' для сносок в конце ноды, либо '`separate`' для сносок в отдельной ноде. См. Раздел 13.10 [Сноски], с. 113.

`@format` Начинает разновидность примера. Похожа на `@example`, но не сужает поля. Парная с `@end format`. См. Раздел 10.3 [@example], с. 87.

`@ftable` *команда-форматирования*
: Начинает двухколоночную таблицу, использующую для каждого вхождения `@item`. Автоматически вносит каждый элемент первой колонки в указатель функций. Парная с `@end ftable`. Аналогична `@table`, за исключением автоматического заполнения указателя функций. См. Раздел 11.3.1 [@ftable и @vtable], с. 97.

`@group` Сохраняет целостность текста, который должен появиться на одной печатной странице. Парная с `@end group`. Не относится к Info. См. Раздел 14.6 [@group], с. 119.

`@H{`*c*`}` Создает длинный венгерский умляут над *c*, как в ő.

`@heading` *название*
Печатает ненумерованный заголовок как для раздела в тексте, но не в содержании печатного руководства. В Info этот заголовок подчеркивается строкой знаков равенства. См. Раздел 5.8 [Команды для разделов], с. 49.

`@headings` *on-off-single-double*
Включает и выключает заголовки, и/или указывает односторонние или двусторонние заголовки страниц нужно печатать. См. Раздел 3.4.6 [Команда @headings], с. 39.

`@html` Полностью переходит в HTML. Парная с `@end html`. См. Раздел 16.3 [Команды прямого форматирования], с. 135.

`@hyphenation{`*пе-ре-не-сен-ное слово*`}`
Явно определяет точки переносов. См. Раздел 14.2 [@- и @hyphenation], с. 118.

`@i{`*текст*`}` Печатает *текст курсивом*. Не относится к Info. См. Раздел 9.2.3 [Шрифты], с. 85.

`@ifclear` *флаг*
Если *флаг* сброшен, команды форматирования Texinfo обрабатывают текст между `@ifclear` *флаг* и следующей командой `@end ifclear`. См. Раздел 16.4 [@set @clear @value], с. 136.

`@ifhtml`
`@ifinfo` Начинает фрагмент текста, который будет проигнорирован TEX, когда он набирает печатное руководство. Этот текст появится только в HTML- или, соотв., в Info-файле. Парная с `@end ifhtml` и, соотв., с `@end ifinfo`. См. Глава 16 [Условно видимый текст], с. 134.

`@ifnothtml`
`@ifnotinfo`
`@ifnottex`
Начинает фрагмент текста, который будет проигнорирован в одном из форматов, но не будет проигнорирован в других. Этот текст появится только в форматах, отличных от заданного. Парная с `@end ifnothtml`, `@end ifnotinfo` или `@end ifnotinfo`, соответственно. См. Глава 16 [Условно видимый текст], с. 134.

`@ifset` *флаг*
Если *флаг* установлен, команды форматирования Texinfo обрабатывают текст между `@ifset` *флаг* и следующей командой `@end ifset`. См. Раздел 16.4 [@set @clear @value], с. 136.

`@iftex` Начинает фрагмент текста, который не появится в Info- и HTML-файле, а будет обработан только TEX. Парная с `@end iftex`. См. Глава 16 [Условно видимый текст], с. 134.

`@ignore` Начинает фрагмент текста, который не появится ни в Info-, ни HTML-файле, ни в печатном руководстве. Парная с `@end ignore`. См. Раздел 1.7 [Комментарии и игнорируемый текст], с. 9.

`@image{`*файл*`, [`*ширина*`], [`*высота*`]}`
Вставляет графическое изображение из внешнего *файла*, масштабированное до заданной *ширины* и/или *высоты*. См. Раздел 13.11 [Рисунки], с. 115.

`@include` *файл*
Вносит содержимое *файла* в Info-файл или печатный документ. См. Приложение E [Включаемые файлы], с. 202.

`@inforef{`*имя-ноды*`, [`*имя-ссылки*`], `*имя-info-файла*`}`
Делает перекрестную ссылку на Info-файл, для которого не существует печатного руководства. См. Раздел 8.7 [Перекрестные ссылки с использованием `@inforef`], с. 73.

`\input` *файл-определений-макросов*
Считывает заданный файл определений макросов. Эта команда используется только в первой строке Texinfo-файла, чтобы TEX считал файл макросов ‘`texinfo`’. Обратная косая черта в `\input` используется вместо `@`, потому что TEX не распознает `@`, пока не прочитает файл с определениями. См. Раздел 3.2 [Заголовок Texinfo-файла], с. 29.

`@item` Обозначает начало помеченного абзаца для `@itemize` и `@enumerate`; обозначает начало текста первой колонки для `@table`, `@ftable` и `@vtable`. См. Глава 11 [Перечни и таблицы], с. 93.

`@itemize` *символ-или-команда-создающая-значок*
Создает последовательность абзацев с отступами и со значком на левом поле в начале каждого абзаца. Парная с `@end itemize`. См. Раздел 11.1 [`@itemize`], с. 93.

`@itemx` Как `@item`, но не создает дополнительный вертикальный пропуск над текстом вхождения. См. Раздел 11.3.2 [`@itemx`], с. 97.

`@kbd{`*символы-клавиатуры*`}`
Обозначает текст, являющийся вводимыми пользователем символами. См. Раздел 9.1.2 [`@kbd`], с. 78.

`@kbdinputstyle` *стиль*
Указывает, должна ли `@kbd` использовать шрифт, отличный от шрифта `@code`. См. Раздел 9.1.2 [`@kbd`], с. 78.

`@key{`*название-клавиши*`}`
Обозначает название клавиши на клавиатуре. См. Раздел 9.1.3 [`@key`], с. 79.

`@kindex` *вхождение*
Добавляет *вхождение* в указатель ключей. См. Раздел 12.1 [Определение вхождений именных указателей], с. 100.

`@L{}`
`@l{}` Создает заглавную и строчную польские перечеркнутые буквы L, соответственно: Ł, ł.

`@lisp` Начинает пример кода на Лиспе. Делает в тексте отступы, не заполняет и выбирает равноширинный шрифт. Парная с `@end lisp`. См. Раздел 10.5 [@lisp], с. 89.

`@lowersections`
Меняет последующие главы на разделы, разделы на подразделы и так далее. См. Раздел 5.12 [@raisesections и @lowersections], с. 51.

`@macro` *имя-макро* {*параметры*}
Определяет новую команду Texinfo `@`*имя-макро*{*параметры*}. Поддерживается только `makeinfo` и `texi2dvi`. См. Раздел 18.1 [Определение макросов], с. 141.

`@majorheading` *название*
Печатает заголовок как для главы в тексте, но не в содержании печатного руководства. Создает больший вертикальный пропуск перед заголовком, чем команда `@chapheading`. В Info этот заголовок подчеркивается строкой звездочек. См. Раздел 5.6 [@majorheading и @chapheading], с. 48.

`@math{`*математическое-выражение*`}`
Форматирует математическое выражение. См. Раздел 13.8 [@math: Вставка математических выражений], с. 110.

`@menu` Отмечает начало меню нод в Info. Не относится к печатному руководству. Парная с `@end menu`. См. Глава 7 [Меню], с. 61.

`@minus{}` Создает знак минус, ‘−’. См. Раздел 13.7 [@minus], с. 110.

`@multitable` *описание-ширины-колонок*
Начинает многоколоночную таблицу. Парная с `@end multitable`. См. Раздел 11.4.1 [Ширина колонок многоколоночных таблиц], с. 98.

`@need` *n* Начинает в печатном руководстве новую страницу, если на текущей осталось меньше *n* мил (тысячных дюйма). См. Раздел 14.7 [@need], с. 120.

`@node` *имя*`,` *следующая*`,` *предыдущая*`,` *вверх*
Определяет начало новой ноды в Info и служит назначением ссылок для TEX. См. Раздел 6.3 [@node], с. 55.

`@noindent`
Предотвращает появление отступов как в новом абзаце для последующего текста. См. Раздел 10.4 [@noindent], с. 88.

`@novalidate`
Подавляет проверку ссылок на ноды, отменяет создание вспомогательных файлов в TEX. Используйте перед `@setfilename`. См. Раздел 20.1.4 [Проверка указателей], с. 162.

`@O{}`
`@o{}` Создает заглавную или строчную перечеркнутые буквы O, соответственно: ffl, fi.

`@oddfooting` [*левая*] `@|` [*средняя*] `@|` [*правая*]
`@oddheading` [*левая*] `@|` [*средняя*] `@|` [*правая*]
: Задает нижние и соотв. верхние заголовки для нечетных (правых) страниц. Допустима только внутри блока '`@iftex`'. См. Раздел F.3 [Как создать свои заголовки], с. 208.

`@OE{}`
`@oe{}`
: Создает заглавную или строчную лигатуры OE, соответственно: ffi, ff. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@option{`*имя-ключа*`}`
: Обозначает ключ командной строки, такой как '`-l`' или '`-help`'. См. Раздел 9.1.9 [`@option`], с. 82.

`@page`
: Начинает в печатном руководстве новую страницу. Не относится к Info. См. Раздел 14.5 [`@page`], с. 119.

`@pagesizes` [*ширина*]`[,` *высота*`]`
: Изменяет размеры страниц. См. Раздел 19.13 [pagesizes], с. 155.

`@paragraphindent` *отступ*
: Делает в абзаце отступ на заданное число пробелов (возможно 0); не меняет отступ, если *отступ* равен `asis`. См. Раздел 3.2.6 [Отступы в абзацах], с. 33.

`@pindex` *вхождение*
: Добавляет *вхождение* в указатель программ. См. Раздел 12.1 [Определение вхождений именных указателей], с. 100.

`@point{}`
: Показывает читателю позицию точки в буфере с помощью знака: '⋆'. См. Раздел 13.9.6 [Обозначение точки в буфере], с. 113.

`@pounds{}`
: Создает символ фунта стерлингов. См. Раздел 13.6 [`@pounds{}`], с. 109.

`@print{}`
: Показывает читателю печатаемый вывод с помощью знака: '⊣'. См. Раздел 13.9.3 [Обозначение печатаемого вывода], с. 111.

`@printindex` *имя-именного-указателя*
: Печатает в печатном руководстве сортированный по алфавиту двухколоночный именной указатель или создает алфавитное меню вхождений именного указателя для Info. См. Раздел 4.1 [Печать именных указателей и меню], с. 43.

`@pxref{`*имя-ноды*`,` [*вхождение*]`,` [*тема-или-название*]`,` [*info-файл*]`,` [*руководство*]`}`
: Создает ссылку, начинающуюся в печатном руководстве словом 'смотрите' со строчной буквы. Используется только в круглых скобках. Не пишите после этой команды знаки препинания — команды форматирования для Info автоматически вставят их, если нужно. Только первый аргумент обязателен. См. Раздел 8.6 [`@pxref`], с. 72.

`@questiondown{}`
: Создает перевернутый вопросительный знак. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@quotation`
: Сужает поля для обозначения текста, цитируемого из другого настоящего или воображаемого произведения. Пишите эту команду на отдельной строке. Парная с `@end quotation`. См. Раздел 10.2 [`@quotation`], с. 87.

`@r{`*текст*`}` Печатает *текст* романским шрифтом. Не влияет на Info. См. Раздел 9.2.3 [Шрифты], с. 85.

`@raisesections`
: Меняет последующие разделы на главы, подразделы на разделы и так далее. См. Раздел 5.12 [`@raisesections` и `@lowersections`], с. 51.

`@ref{`*имя-ноды*`, [`*вхождение*`], [`*тема-или-название*`], [`*info-файл*`], [`*руководство*`]}`
: Создает ссылку. В печатном руководстве ссылка не начинается словом ‘Смотрите’. Ставьте после этой команды знак препинания. Только первый аргумент обязателен. См. Раздел 8.5 [`@ref`], с. 71.

`@refill` В Info, перезаполняет абзац и делает в нем отступ после того, как закончена остальная обработка. Не влияет на TEX, который перезаполняет всегда. В этой команде больше нет необходимости, так как все форматирующие программы теперь перезаполняют автоматически. См. Приложение H [Перезаполнение абзацев], с. 219.

`@result{}`
: Показывает читателю результат выражения с помощью специального знака: ‘⇒’. См. Раздел 13.9.1 [`@result`], с. 111.

`@ringaccent{`*c*`}`
: Создает акцент кружок над следующим символом, как в o. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@samp{`*текст*`}`
: Выделяет *текст*, являющийся буквальным примером последовательности символов. Используется для одиночных символов, операторов и часто для полных команд оболочки. См. Раздел 9.1.4 [`@samp`], с. 80.

`@sc{`*текст*`}`
: Набирает *текст* в печатном выводе ШРИФТОМ МАЛЕНЬКИХ ЗАГЛАВНЫХ БУКВ, а в Info-файле — заглавными буквами. См. Раздел 9.2.2 [Маленькие заглавные буквы], с. 84.

`@section` *название*
: Начинает раздел внутри главы. В печатном руководстве заголовок этого раздела нумеруется и вносится в содержание. В Info этот заголовок подчеркивается строкой знаков равенства. См. Раздел 5.7 [`@section`], с. 49.

`@set` *флаг* [*строка*]
: Делает *флаг* активным, заставляя команды форматирования Texinfo обрабатывать текст между последовательными парами команд `@ifset` *флаг* и `@end ifset`. Возможно, устанавливает значение *флага* равным *строке*. См. Раздел 16.4 [`@set @clear @value`], с. 136.

`@setchapternewpage` *on-off-odd*

Указывает, должны ли главы начинаться на новой странице, и, если да, то должны ли это быть нечетные (правые) страницы. См. Раздел 3.2.5 [@setchapternewpage], с. 32.

`@setcontentsaftertitlepage`

Помещать содержание после '`@end titlepage`', даже если `@contents` находится в другом месте. См. Раздел 4.2 [Содержание], с. 44.

`@setfilename` *имя-info-файла*

Предоставляет имя для Info-файла. Эта команда также важна и для форматирования с TEX, хотя и не производит никакого вывода. См. Раздел 3.2.3 [@setfilename], с. 30.

`@setshortcontentsaftertitlepage`

Помещать краткое содержание после команды '`@end titlepage`', даже если команда `@shortcontents` находится в другом месте. См. Раздел 4.2 [Содержание], с. 44.

`@settitle` *название*

Предоставляет название для заголовков страниц в печатном руководстве. См. Раздел 3.2.4 [@settitle], с. 31.

`@shortcontents`

Печатает краткое содержание. Не относится к Info, которая использует меню, а не содержание. Синоним `@summarycontents`. См. Раздел 4.2 [Создание содержания], с. 44.

`@shorttitlepage` *название*

Создает минимальный титульный лист. См. Раздел 3.4.1 [@titlepage], с. 35.

`@smallbook`

Заставляет TEX выводить печатное руководстве в формате 7 на 9.25 дюймов, а не в обычном формате 8.5 на 11 дюймов. См. Раздел 19.11 [Печать маленьких книг], с. 154. Также, смотрите Раздел 10.6 [small], с. 89.

`@smalldisplay`

Начинает разновидность примера. Как `@smallexample` (делает в тексте отступы, не заполняет), но не переключается на равноширинный шрифт. В формате `@smallbook`, печатает текст меньшим шрифтом, чем `@display`. Парная с `@end smalldisplay`. См. Раздел 10.6 [small], с. 89.

`@smallexample`

Делает в тексте отступ для обозначения примера. Не заполняет, переключается на равноширинный шрифт. В формате `@smallbook`, печатает текст меньшим шрифтом, чем `@example`. Парная с `@end smallexample`. См. Раздел 10.6 [small], с. 89.

`@smallformat`

Начинает разновидность примера. Как `@smalldisplay`, но не сужает поля и не переключается на равноширинный шрифт. В формате `@smallbook`,

печатает текст меньшим шрифтом, чем `@format`. Парная с `@end smallformat`. См. Раздел 10.6 [small], с. 89.

`@smalllisp`
: Начинает пример кода на Лиспе. Делает в тексте отступ, не заполняет, переключается на равноширинный шрифт. В формате `@smallbook`, печатает текст меньшим шрифтом. Парная с `@end smalllisp`. См. Раздел 10.6 [small], с. 89.

`@sp` *n*
: Пропускает *n* пустых строк. См. Раздел 14.4 [@sp], с. 119.

`@ss{}`
: Создает немецкую букву острый S или эс-цет, ı. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@strong {`*текст*`}`
: Обозначает логическое ударение, падающее на *текст*, набирая этот текст **жирным** шрифтом для печатного руководства и окружая его звездочками для Info. См. Раздел 9.2.1 [Логическое ударение], с. 84.

`@subheading` *название*
: Печатает заголовок, как для подраздела, в тексте, но не в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой дефисов. См. Раздел 5.10 [@unnumberedsubsec @appendixsubsec @subheading], с. 50.

`@subsection` *название*
: Начинает подраздел внутри раздела. В печатном руководстве, заголовок подраздела нумеруется и вносится в содержание. В Info этот заголовок подчеркивается строкой дефисов. См. Раздел 5.9 [@subsection], с. 50.

`@subsubheading` *название*
: Печатает заголовок как для подподраздела в тексте, но не в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой точек. См. Раздел 5.11 [Команды ‘subsub’], с. 50.

`@subsubsection` *название*
: Начинает подподраздел внутри подраздела. В печатном руководстве, заголовок подподраздела нумеруется и вносится в содержание. В Info этот заголовок подчеркивается строкой точек. См. Раздел 5.11 [Команды ‘subsub’], с. 50.

`@subtitle` *название*
: В печатном руководстве, набирает подзаголовок шрифтом обычного размера, прижатый к правому краю страницы. Не относится к Info, в которой нет титульных листов. См. Раздел 3.4.3 [Команды @title @subtitle и @author], с. 37.

`@summarycontents`
: Печатает краткое содержание. Не относится к Info, в которой вместо содержания используются меню. Синоним `@shortcontents`. См. Раздел 4.2 [Создание содержания], с. 44.

`@syncodeindex` *источник назначение*
: Вносит именной указатель, заданный первым аргументом, в указатель, заданный вторым аргументом, печатает вхождения первого указателя шрифтом `@code`. См. Раздел 12.4 [Объединение именных указателей], с. 102.

`@synindex` *источник назначение*
: Вносит именной указатель, заданный первым аргументом, в указатель, заданный вторым аргументом. Не изменяет шрифт вхождений *источника*. См. Раздел 12.4 [Объединение именных указателей], с. 102.

`@t{`*текст*`}` Печатает *текст* `равноширинным` шрифтом, в стиле печатной машинки. Не влияет на Info. См. Раздел 9.2.3 [Шрифты], с. 85.

`@tab` Разделяет колонки в многоколоночной таблице. См. Раздел 11.4.2 [Строки многоколоночных таблиц], с. 99.

`@table` *форматирующая-команда*
: Начинает двухколоночную таблицу, использующую для каждого вхождения `@item`. Пишите каждое вхождение первой колонки на той же строке, что и `@item`. Вхождения первой колонки печатаются шрифтом, определяемым *форматирующей-командой*. Парная с `@end table`. См. Раздел 11.3 [Создание двухколоночных таблиц], с. 96. Также смотрите Раздел 11.3.1 [`@ftable` и `@vtable`], с. 97, Раздел 11.3.2 [`@itemx`], с. 97.

`@TeX{}` Вставляет лого TEX. См. Раздел 13.5 [Вставка знаков TEX и ©], с. 109.

`@tex` Полностью переключает в TEX. Парная с `@end tex`. См. Раздел 16.3 [Команды прямого форматирования], с. 135.

`@thischapter`
`@thischaptername`
`@thisfile`
`@thispage`
`@thistitle`
: Допустимы только в верхних или нижних заголовках. Обозначают номер и название текущей главы (в формате ‘Глава 1: Название’), только название главы, имя файла, номер текущей страницы и название документа, соответственно. См. Раздел F.3 [Как создать свои заголовки], с. 208.

`@tieaccent{`*cc*`}`
: Создает акцент лигу над следующими двумя символами *cc*, как в ‘o͡o’. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@tindex` *вхождение*
: Добавляет *вхождение* в указатель типов данных. См. Раздел 12.1 [Определение вхождений именных указателей], с. 100.

`@title` *название*
: В печатном руководстве, набирает заголовок прижатым к левому краю страницы шрифтом, большим, чем обычный, и подчеркивает его черной линейкой. Не относится к Info, в которой нет титульных листов. См. Раздел 3.4.3 [Команды `@title` `@subtitle` и `@author`], с. 37.

`@titlefont{`*текст*`}`
: В печатном руководстве, печатает *текст* шрифтом, большим, чем обычный. Не относится к Info, в которой нет титульных листов. См. Раздел 3.4.2 [Команды @titlefont @center и @sp], с. 36.

`@titlepage`
: Указывает Texinfo на начало титульного листа. Пишите эту команду на отдельной строке. Парная с `@end titlepage`. Текст между `@titlepage` и `@end titlepage` не появляется в Info. См. Раздел 3.4.1 [@titlepage], с. 35.

`@today{}`
: Вставляет текущую дату в стиле ‘1 янв 1900’. См. Раздел F.3 [Как создать свои заголовки], с. 208.

`@top` *название*
: Если Texinfo-файл будет обрабатываться с помощью `makeinfo`, указывает самую первую строку `@node` в файле, которая должна быть написана непосредственно перед командой `@top`. Используется для способности `makeinfo` вставлять указатели. Этот заголовок подчеркивается строкой звездочек. И строка `@node`, и строка `@top` обычно должны быть окружены `@ifinfo` и `@end ifinfo`. В TEX и `texinfo-format-buffer`, команда `@top` просто синоним для `@unnumbered`. См. Раздел 6.4 [Создание указателей с помощью makeinfo], с. 59.

`@u{`*c*`}`
`@ubaraccent{`*c*`}`
`@udotaccent{`*c*`}`
: Создают акцент краткости, подчерк или точку снизу, соответственно, над или под символом *c*, как в ŏ, o̲, ọ. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@unnumbered` *название*
: В печатном руководстве начинает главу появляющуюся без какого-либо номера. Ее заголовок появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой звездочек. См. Раздел 5.5 [@unnumbered и @appendix], с. 48.

`@unnumberedsec` *название*
: В печатном руководстве начинает раздел появляющийся без какого-либо номера. Его заголовок появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой знаков равенства. См. Раздел 5.8 [Команды для разделов], с. 49.

`@unnumberedsubsec` *название*
: В печатном руководстве начинает подраздел появляющийся без какого-либо номера. Его заголовок появляется в содержании печатного руководства. В Info, этот заголовок подчеркивается строкой дефисов. См. Раздел 5.10 [@unnumberedsubsec @appendixsubsec @subheading], с. 50.

`@unnumberedsubsubsec` *название*
: В печатном руководстве начинает подподраздел появляющийся без какого-либо номера. Его заголовок появляется в содержании печатного руковод-

ства. В Info, этот заголовок подчеркивается строкой точек. См. Раздел 5.11 [Команды subsub], с. 50.

`@uref{`*url*`[, `*отображаемый-текст*`][`*замещение*`]}`
Определяет перекрестную ссылку на внешний унифицированный указатель ресурса для World Wide Web. См. Раздел 8.8 [@url], с. 74.

`@url{`*url*`}` Обозначают текст, являющийся унифицированным указателем ресурса для World Wide Web. См. Раздел 9.1.13 [@url], с. 83.

`@v{`*c*`}` Создает галочку над следующим символом *c*, как в ǒ. См. Раздел 13.3 [Вставка акцентов], с. 108.

`@value{`*флаг*`}`
Замещает *флаг* значением, в которое он было установлен командой `@set` *флаг*. См. Раздел 16.4 [@set @clear @value], с. 136.

`@var{`*метасинтаксическая-переменная*`}`
Выделяет метасинтаксическую переменную, то есть нечто, обозначающее другой кусок текста. См. Раздел 9.1.5 [Обозначение метасинтаксических переменных], с. 80.

`@vindex` *вхождение*
Добавляет *вхождение* в указатель переменных. См. Раздел 12.1 [Определение вхождений именных указателей], с. 100.

`@vskip` *количество*
В печатном руководстве вставляет пропуск, так, чтобы текст на оставшейся части страницы передвинулся ниже. Используется при форматировании страницы с информацией об авторских правах с аргументом ‘`0pt plus 1filll`’. (Обратите внимание на написание ‘`filll`’.) `@vskip` может быть использована только в контексте, игнорируемом Info. См. Раздел 3.4.4 [Авторские права и разрешения], с. 38.

`@vtable` *форматирующая-команда*
Начинает двухколоночную таблицу, использующую для каждого вхождения `@item`. Автоматически вносит каждый элемент первой колонки в указатель переменных. Парная с `@end vtable`. Аналогична `@table`, за исключением автоматического заполнения указателя переменных. См. Раздел 11.3.1 [@ftable и @vtable], с. 97.

`@w{`*текст*`}` Предотвращает разбиение *текста* между двух строк. Не завершайте абзац, который использует `@w` командой `@refill`. См. Раздел 14.3 [@w], с. 118.

`@xref{`*имя-ноды*`, [`*вхождение*`], [`*тема-или-название*`], [`*info-файл*`], [`*руководство*`]}`
Делает ссылку, начинающуюся в печатном руководстве словом ‘Смотрите’. Ставьте после этой команды знак препинания. Только первый аргумент обязателен. См. Раздел 8.3 [@xref], с. 67.

# Приложение B Советы и подсказки

Вот несколько советов по написанию документации в Texinfo:

- Пишите в настоящем времени, а не в прошедшем или будущем.
- Пишите активно! Например, пишите “Мы рекомендуем . . .” вместо “Рекомендуется . . .”.
- Пишите в 70 или 72 колонки. Более длинные строки трудно читать.
- Включайте уведомление об авторских правах и разрешение на копирование.

## Больше именных указателей!

Пишите много вхождений в именные указатели, различными способами. Читателям нравятся именные указатели; они полезны и удобны.

Хотя проще писать вхождения именных указателей в процессе написания тела текста, некоторые предпочитают писать их после. В любом случае, пишите вхождения перед абзацем, к которому они относятся. Тогда вхождение будет указывать на первую страницу абзаца, который разделен между страницами.

Вот еще несколько советов, которые мы сочли полезными:

- Пишите каждое вхождение по-своему, чтобы каждое ссылалось на отдельное место в документе.
- Пишите вхождения только там, где эта тема обсуждается подробно. Например, нет смысла включать в указатель понятий вхождение “отладочная информация” в главе о том, как сообщить авторам программы о найденных вами ошибках. Кто-то, кто хочет узнать об отладочной информации определенно не найдет ничего в этой главе.
- Всегда пишите первое слово каждого вхождения с заглавной буквы, либо всегда со строчной. Короткие вхождения часто призывают к использованию строчных букв, а более длинные вхождения – заглавных. Какого бы соглашения по использованию заглавных букв вы не придерживались, пожалуйста, делайте это всегда! Смешивание двух стилей смотрится плохо.
- Всегда используйте заглавные буквы в тех словах в именном указателе, для которых это будет правильно, таких как названия стран или аббревиатуры. Всегда используйте правильный регистр в регистрозависимых названиях, таких как Си или Лисп.
- Пишите команды для именных указателей, относящихся к целому разделу, сразу после команды, создающей этот раздел, а для именных указателей, относящихся к абзацу, — перед этим абзацем.

  В нижеприведенном примере после вхождения для “Чаепития” идет пустая строка:

```
@section Чай и булки
@cindex Завтрак
@cindex Чаепитие

@cindex Булки, французские, к чаю
@cindex Французские булки, к чаю
@cindex Чай, с булками
Съешь ещё этих мягких французских булок да выпей чаю.
```

(Заметьте, что этот пример показывает вхождения для одного понятия, записанные разными способами — ‘`Булки, французские`’ и ‘`Французские булки`’ — так что читатели могут найти их в именном указателе по-разному.)

## Пустые строки

- Вставляйте пустую строку между командой описания структуры глав и первым предложением последующего абзаца или между командами создания вхождений именных указателей, связанных с командой структурирования, и первым предложением последующего абзаца, как показано в примере об именных указателях. Иначе программа форматирования выведет заголовок и абзац слишком близко друг к другу.
- Всегда вставляйте пустую строку перед командой `@table` и после команды `@end table`; но никогда не вставляйте после команды `@table` или перед `@end table`.

  Например,

```
Разновидности булок:

@table @samp
@item Мягкие
Хороши к чаю.

@item Французские
Тоже хороши к чаю.
@end table

@noindent
С другой стороны, ...
```

  Вставляете пустые строки до и после `@itemize ... @end itemize` и `@enumerate ... @end enumerate` таким же образом.

## Завершайте фразы

Завершенные фразы легче читать, чем ...

- Пишите элементы перечней как полные предложения или по крайней мере как завершенные фразы. Неполные выражения ... неуклюжи ... это.
- Пишите вводные предложения или фразы для многоэлементных перечней или таблиц как законченные выражения. Не пишите “Вы можете установить:”; вместо этого пишите “Вы можете установить такие переменные:”. Первое выражение звучит как урезанное.

## Редакции, даты и версии

Пишите дату и номера редакции и версии в трех местах в каждом руководстве:

1. В первом разделе `@ifinfo`, для читающих Texinfo-файл.
2. В разделе `@titlepage`, для читающих печатное руководство.
3. В первой ноде , для читающих Info-файл.

Также полезно написать заметку перед первым разделом `@ifinfo` для объяснения, что вы делаете.

Например:

```
@c ===> ВНИМАНИЕ! <==
@c Укажите дату и номера редакции и версии
@c в *трех* местах:
@c   1. Первый раздел ifinfo   2. титульный лист  3. первая нода
@c Чтобы найти эти места, ищите !!установить

@ifinfo
@c !!установить редакцию, дату, версию
Редакция 4.03, @cite{Руководство по GDB} для GDB версии 4.3,
январь 1992 г.

...
```

— или используйте `@set` и `@value` (см. Раздел 16.4.3 [Пример `@value`], с. 138).

## Команды для определений

Команды для определений, а это `@deffn`, `@defun`, `@defmac` и им подобные, позволяют вам делать описания в едином формате.

- Пишите ровно одну команду определения для каждого определяемого элемента. При этом автоматически создается вхождение именного указателя, приводящее читателя к определению.
- В приложении, содержащем обзор функций, используйте `@table ... @end table`, а не `@deffn` или другие команды для определений.

## Заглавные буквы

- Пишите “Texinfo” с заглавной буквы; это название. Не делайте ‘`x`’ или ‘`i`’ заглавными.
- Пишите “Info” с заглавной буквы; это название.
- Пишите TEX, используя команду `@TeX{}`. Обратите внимание на заглавные ‘`T`’ и ‘`X`’. Эта команда заставляет программы форматирования набирать это название в соответствии с пожеланиями Дональда Кнута, автора TEX.

## Пробелы

Не используйте для форматирования Texinfo-файла пробелы, за исключением текста внутри `@example ... @end example` и похожих блоков.

Например, TEX заполняет следующее:

```
@kbd{C-x v}
@kbd{M-x vc-next-action}
   Производит следующую логическую операцию
   над файлом с контролем версий,
   соответствующим текущему буферу.
```

таким образом, что оно выглядит так:

> `C-x v M-x vc-next-action` Производит следующую логическую операцию над файлом с контролем версий, соответствующим текущему буферу.

В этом случае, текст нужно форматировать с помощью `@table`, `@item` и `@itemx`, для создания таблицы.

### `@code`, `@samp`, `@var` и '`--`'

- Используйте `@code` вокруг символов Лисп, включая имена команд. Например,

  ```
  Главная функция -- это @code{vc-next-action}, ...
  ```

- Не помещайте буквы вроде '`s`' сразу после '`@code`'. Такие буквы смотрятся плохо.
- Используйте `@var` вокруг мета-переменных. Не окружайте их угловыми скобками.
- Используйте три дефиса подряд , '`--`', для обозначения тире. TEX набирает их как тире, а программы форматирования для Info сокращают три дефиса до двух.

### Точки вне кавычек

Помещайте точки и другие знаки препинания *вне* кавычек, если только эти знаки препинания не относятся к словам, заключенным в кавычки. Эта практика противоречит издательским соглашениям, принятым в Соединенных Штатах[1], но позволяет читателю различать содержимое кавычек и целого отрывка.

Например, вы должны написать точку в следующем предложении вне кавычек:

```
Очевидно, 'au' является сокращением от ■author■.
```

так как '`au`' *не* служит сокращением от '`author.`' (с точкой после слова).

### Введение новых терминов

- Вводите новые термины так, чтобы читатель, не знакомый с ними, мог понять их из контекста; или напишите определение термина.

  Например, в следующем примере термины "check in", "register" и "delta" появляются впервые; это предложение должно быть переписано, чтобы они были понятны.

  > The major function assists you in checking in a file to your version control system and registering successive sets of changes to it as deltas.

- Используйте команду `@dfn` вокруг вводимого слова, чтобы обозначить, что у читателя не предполагается знание его смысла, и что он может сможет узнать его значение из данного отрывка.

### `@pxref`

Абсолютно никогда не используйте `@pxref`, кроме как в особом контексте, для которого она была разработана: внутри круглых скобок, когда закрывающая круглая скобка идет сразу после закрывающей фигурной. Одна форматирующая программа автоматически вставляет закрывающие знаки препинания, а вторая — нет. Это означает, что вывод выглядит правильно и на печати, и в Info-файле, но только если эта команда применена внутри круглых скобок.

---

[1] Но соответствует правилам пунктуации русского языка. *(Прим. переводчика)*

## Вызов из оболочки

Вы можете вызвать такие программы, как Emacs, GCC и `gawk` из оболочки. Документация для любой программы должна содержать раздел, описывающий это. К сожалению, если имена нод и названия этих разделов всегда различны, читатели считают, что найти этот раздел трудно.

Называйте такие разделы фразой, начинающейся словом ‘Вызов . . .’, например ‘Вызов Emacs’; тогда пользователи смогут легко найти этот раздел.

## Синтаксис ANSI C

Когда вы используете `@example` для описания соглашений о вызове функции Си, используйте синтаксис ANSI C, как здесь:

```
void dld_init (char *@var{path});
```

И в последующем обсуждении ссылайтесь на значения аргументов по тем же именам аргументов, снова выделяя их с помощью `@var`.

Избегайте использования устаревшего синтаксиса, выглядящего так:

```
#include <dld.h>

dld_init (path)
char *path;
```

Также, лучше избегать написания `#include` над объявлением лишь для того, чтобы показать, что эта функция объявлена в заголовочном файле. Это может произвести неправильное впечатление, что `#include` относится к объявлению функции. Либо скажите явно, какой заголовочный файл содержит объявление, либо, что еще лучше, напишите имя заголовочного файла, используемого для группы функций, в начале раздела, описывающего эти функции.

## Плохие примеры

Вот несколько примеров плохого стиля, которого нужно избегать:

В этом примере скажите “ . . . you must `@dfn{`check in`}` the new version.” Это более гладко читается.

> When you are done editing the file, you must perform a `@dfn{`check in`}`.

В следующем примере скажите “. . . makes a unified interface such as VC mode possible.”

> SCCS, RCS and other version-control systems all perform similar functions in broadly similar ways (it is this resemblance which makes a unified control mode like this possible).

И в этом примере, вам нужно указать, к чему относится ‘it’:

> If you are working with other people, it assists in coordinating everyone’s changes so they do not step on each other.

## И наконец ...

- Произносите TEX, как если бы буква ‘X’ была греческой буквой ‘хи’, как последний звук в имени ‘Бах’. Но Texinfo произносите как в ‘speck’: “текинфо”.
- Пишите заметки для себя в самом конце Texinfo-файла после `@bye`. Ни одна из программ форматирования не обрабатывает текст после `@bye`; это так же, как если бы текст был между `@ignore ... @end ignore`.

# Приложение C Пример Texinfo-файла

Это пример законченного короткого Texinfo-файла, без каких-либо комментариев. Вы можете найти этот пример с комментариями в первой главе. См. Раздел 1.10 [Короткий пример Texinfo-файла], с. 11.

```
\input texinfo   @c -*-texinfo-*-
@c %**start of header
@setfilename sample.info
@settitle Пример документа
@c %**end of header

@setchapternewpage odd

@ifinfo
Это короткий пример законченного Texinfo-файла.

Copyright @copyright{} 1999 Free Software Foundation, Inc.
@end ifinfo

@titlepage
@sp 10
@comment Заголовок печатается крупным шрифтом
@center @titlefont{Пример заголовка}

@c   Две следующие команды начинают страницу с
@c   информацией об авторских правах
@page
@vskip 0pt plus 1filll
Copyright @copyright{} 1999 Free Software Foundation, Inc.
@end titlepage

@node    Top,      Первая глава,           , (dir)
@comment имя-ноды, следующая,    предыдущая, вверх

@menu
* Первая глава::       Первая и единственная глава в
                       этом примере.
* Указатель понятий::  В этом указателе есть два элемента.
@end menu

@node    Первая глава , Указатель понятий, Top    , Top
@comment имя-ноды,      следующая,     предыдущая, вверх
@chapter Первая глава
@cindex Пример вхождения именного указателя

Это содержимое первой главы.
@cindex Другой пример вхождения именного указателя.
```

```
Это нумерованный перечень.

@enumerate
@item
Это первый пункт.

@item
Это второй пункт.
@end enumerate

Команды @code{makeinfo} и @code{texinfo-format-buffer}
преобразуют Texinfo-файлы, такие как этот, в Info-файлы,
а @TeX{} форматирует печатное руководство.

@node    Указатель понятий,          , Первая глава, Top
@comment имя-ноды         , следующая, предыдущая  , вверх
@unnumbered Указатель понятий

@printindex cp

@contents
@bye
```

# Приложение D Пример разрешений

Texinfo-файлы должны содержать разделы, сообщающие читателям, что они имеют право копировать и распространять Texinfo-файл, Info-файл и печатное руководство. Также, если вы пишите руководство по программному обеспечению, вы должны объяснить, что это программное обеспечение свободно и либо включить Универсальную Общественную Лицензию GNU (GPL), либо предоставить ссылку на нее. См. раздел "Distribution" в *Руководство по GNU Emacs*, – это пример текста, который можно использовать в разделах документа "Распространение", "Универсальная Общественная Лицензия" и "ГАРАНТИЙ НЕТ". См. [Условия копирования Texinfo], с. 2, для получения примера краткого объяснения, как условия копирования наделяют вас правами.

В Texinfo-файле, первый раздел `@ifinfo` обычно начинается со строки, говорящей, что документирует этот файл. Это то, что первым увидит человек, читающий необработанный Texinfo-файл или использующий продвинутую команду Info *g **. См. Info файл '`info`', node '`Expert`', для дальнейшей информации. (Читатель, использующий обычные команды Info, как правило начинает читать с первой ноды и пропускает этот первый раздел, не входящий ни в одну ноду.)

В разделе `@ifinfo` после обзорного предложения следует уведомление об авторских правах, а за ним уведомление о разрешении на копирование. Один из абзацев разрешений на копирование окружается командами `@ignore` и `@end ignore`. Этот абзац говорит, что Texinfo-файл разрешается обрабатывать с помощью TEX и печатать, если печатное руководство содержит соответствующее уведомление об авторских правах. Этот абзац не вносится в Info-файл, так как он не относится к Info-файлу; но он является обязательной частью Texinfo-файла, так как разрешает людям обрабатывать этот Texinfo-файл в TEX и печатать результаты.

В печатном руководстве уведомление о предоставляемом Фондом Свободного Программного Обеспечения разрешении на копирование следует после уведомления об авторских правах и издательских сведений и располагается в области, очерченной командами `@titlepage` и `@end titlepage`. Уведомление о разрешении на копирование точно то же, что и уведомление в разделе `@ifinfo`, за исключением того, что абзац, заключенный между `@ignore` и `@end ignore` не является частью этого уведомления.

Чтобы проще было вставить уведомления о разрешениях в каждый раздел Texinfo-файла, образец уведомлений для каждого раздела приведены ниже полностью.

Вам может понадобиться указать правильное название раздела, упомянутого в уведомлении о разрешениях. Например, в *Руководстве по GDB*, раздел, ссылающийся на Универсальную Общественную Лицензию называется "Универсальная Общественная Лицензия GDB", но в примере, показанном ниже, на этот раздел ссылаются как обычно, "Универсальная Общественная Лицензия GNU". Если Texinfo-файл не содержит копию Универсальной Общественной Лицензии, исключите ссылку на нее, но оставьте конец предложения.

## D.1 Разрешения на копирование для Info

В разделе `@ifinfo` Texinfo-файла, стандартное для Фонда Свободного Программного Обеспечения уведомление об авторских правах выглядит следующим образом:

```
Этот файл описывает ...

Copyright 1998 Free Software Foundation, Inc.



Permission is granted to make and distribute verbatim
copies of this manual provided the copyright notice and
this permission notice are preserved on all copies.

@ignore
Permission is granted to process this file through TeX
and print the results, provided the printed document
carries a copying permission notice identical to this
one except for the removal of this paragraph (this
paragraph not being relevant to the printed manual).

@end ignore
Permission is granted to copy and distribute modified
versions of this manual under the conditions for
verbatim copying, provided also that the sections
entitled ■Copying■ and ■GNU General Public License■
are included exactly as in the original, and provided
that the entire resulting derived work is distributed
under the terms of a permission notice identical to this
one.

Permission is granted to copy and distribute
translations of this manual into another language,
under the above conditions for modified versions,
except that this permission notice may be stated in a
translation approved by the Free Software Foundation.
```

## D.2 Разрешение на копирование на титульном листе

В разделе Texinfo-файла `@titlepage` после уведомления об авторских правах и издательских сведений следует стандартное для Фонда Свободного Программного Обеспечения уведомление о разрешении на копирование. Стандартная формулировка выглядит так:

```
Permission is granted to make and distribute verbatim
copies of this manual provided the copyright notice and
this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified
versions of this manual under the conditions for
verbatim copying, provided also that the sections
entitled ■Copying■ and ■GNU General Public License■
```

```
are included exactly as in the original, and provided
that the entire resulting derived work is distributed
under the terms of a permission notice identical to this
one.

Permission is granted to copy and distribute
translations of this manual into another language,
under the above conditions for modified versions,
except that this permission notice may be stated in a
translation approved by the Free Software Foundation.
```

# Приложение E Включаемые файлы

Когда TEX или команда форматирования Info встречают в Texinfo-файле команду `@include`, они обрабатывают содержимое указанного командой файла и включают его в создаваемый DVI- или Info-файл. Вхождения именных указателей из включаемых файлов вносятся в именные указатели выходного файла.

Включаемые файлы позволяют вам держать большой документ в виде удобных небольших частей.

## E.1 Как использовать включаемые файлы

Чтобы включить в Texinfo-файл другой файл, напишите команду `@include` в начале строки, и за ней имя включаемого файла на той же строке. Например:

```
@include buffers.texi
```

Включаемый файл должен быть просто сегментом текста, который вы предполагаете вставить как есть в общий или *внешний* Texinfo-файл; он не должен содержать стандартные начальную и завершающую части Texinfo-файла. В особенности, вы не должны начинать включаемый файл строкой '`\input texinfo`'; если вы сделаете так, эта фраза вставится во внешний файл буквально. Аналогично, вы не должны завершать включаемый файл командой `@bye`; текст после `@bye` не форматируется.

Раньше в начале каждого включаемого файла нужно было обязательно писать строку `@setfilename`, но теперь не нужно. Теперь не имеет значения, написали ли вы такую строку или нет. Если строка `@setfilename` существует во включаемом файле, она игнорируется.

По соглашению, включаемый файл начинается строкой `@node`, за которой следует строка `@chapter`. Каждый включаемый файл — это одна глава. Это облегчает использование обычных команд создания и обновления нод и меню внутри включаемого файла. Однако, простые команды Emacs для создания и обновления нод и меню не работают со множественными Texinfo-файлами. То есть вы не можете использовать эти команды для заполнения указателей 'Next', 'Previous' и 'Up' в строке `@node`, которая начинает включаемый файл. Также вы не можете использовать обычные команды для создания главного меню полного файла. Вы либо должны заполнить меню и указатели 'Next', 'Previous' и 'Up' вручную, либо использовать команду режима Texinfo для GNU Emacs, `texinfo-multiple-files-update`, разработанную для `@include`-файлов.

## E.2 `texinfo-multiple-files-update`

Режим Texinfo в GNU Emacs предоставляет команду `texinfo-multiple-files-update`. Эта команда создает или обновляет во включаемом файле указатели 'Next', 'Previous' и 'Up', а также во внешнем или общем Texinfo-файле; и создает или обновляет главное меню во внешнем файле. В зависимости от того, вызвали ли вы ее с необязательными аргументами, эта команда обновляет либо только указатели в первой строке `@node` включаемого файла, либо все указатели:

*`M-x texinfo-multiple-files-update`*
: Вызванная без аргументов:

- Создает или обновляет указатели ‘Next’, ‘Previous’ и ‘Up’ в первой строке `@node` в каждом файле, включенном во внешний или общий Texinfo-файл.
- Создает или обновляет указатели первого уровня во внешнем или общем файле.
- Создает или обновляет главное меню во внешнем файле.

*C-u M-x texinfo-multiple-files-update*

Вызванная с *C-u* в качестве префиксного аргумента:

- Создает или обновляет указатели в первой строке `@node` в каждом включаемом файле.
- Создает или обновляет указатели первого уровня во внешнем файле.
- Создает и вставляет главное меню во внешнем файле. Главное меню составляется из всех меню всех включаемых файлов.

*C-u 8 M-x texinfo-multiple-files-update*

Вызванная с числовым префиксным аргументом, например *C-u 8*:

- Создает или обновляет **все** указатели ‘Next’, ‘Previous’ и ‘Up’ во всех включаемых файлах.
- Создает или обновляет **все** меню во всех включаемых файлах.
- Создает или обновляет указатели первого уровня во внешнем или общем файле.
- И затем создает главное меню во внешнем файле. Это похоже на вызов `texinfo-master-menu` с аргументом, когда вы работаете только с одним файлом.

Обратите внимание на применение префиксного аргумента при интерактивном использовании: с обычным префиксным аргументом, просто *C-u*, команда `texinfo-multiple-files-update` вставляет главное меню; с числовым префиксным аргументом, таким как *C-u 8*, эта команда обновляет **все** указатели и меню во **всех** файлах и затем вставляет главное меню.

## E.3 Требования для включаемых файлов

Если вы планируете использовать команду `texinfo-multiple-files-update`, то внешний Texinfo-файл, перечисляющий включаемые файлы, не должен содержать ничего, кроме начальной и завершающей частей Texinfo-файла и нескольких команд `@include` для включаемых файлов. Он не должен включать даже именных указателей, которые должны быть перечислены в отдельном включаемом файле.

Кроме того, каждый включаемый файл должен содержать ровно одну ноду верхнего уровня (обычно это `@chapter` или эквивалент), и эта ноды должна быть первой нодой во включаемом файле. Более того, все эти нод верхнего уровня из каждого включаемого файла должны быть на одном иерархическом уровне в общей структуре. Обычно все они являются нодами `@chapter`, `@appendix` или `@unnumbered`. Таким образом, каждый включаемый файл содержит одну и только одну ноду с главой или эквивалентного уровня.

Внешний файл должен содержать только *одну* ноду, ноду ‘Top’. Он не должен содержать *никаких* нод кроме одной ноды ‘Top’. Команда `texinfo-multiple-files-update` не будет их обрабатывать.

## E.4 Пример файла с `@include`

Вот пример законченного внешнего Texinfo-файла с `@include`-файлами внутри, до запуска `texinfo-multiple-files-update`, которая вставила бы главное меню:

```
\input texinfo @c -*-texinfo-*-
@setfilename  include-example.info
@settitle Include Example

@setchapternewpage odd
@titlepage
@sp 12
@center @titlefont{Include Example}
@sp 2
@center by Whom Ever

@page
@vskip 0pt plus 1filll
Copyright @copyright{} 1998 Free Software Foundation, Inc.
@end titlepage

@ifinfo
@node Top, First, , (dir)
@top Master Menu
@end ifinfo

@include foo.texinfo
@include bar.texinfo
@include concept-index.texinfo

@summarycontents
@contents


@bye
```

Включаемый файл, например ‘`foo.texinfo`’, может выглядеть так:

```
@node First, Second, , Top
@chapter First Chapter

Contents of first chapter ...
```

Полное содержание ‘`concept-index.texinfo`’ может выглядеть так:

```
@node Concept Index
@unnumbered Concept Index


@printindex cp
```

Внешний исходный Texinfo-файл для *The GNU Emacs Lisp Reference Manual* называется ‘`elisp.texi`’. Этот внешний файл содержит главное меню из 417 пунктов и список из 41 включаемого файла.

## E.5 История развития включаемых файлов

Когда Info была первоначально создана, было принято делать много небольших Info-файлов на одну тему. Каждый Info-файл форматировался из отдельного исходного Texinfo-файла. Этот обычай был предназначен для того, чтобы в Emacs не нужно было создавать большой буфер для хранения большого Info-файла целиком, когда кто-то хотел получить информацию; вместо этого Emacs выделял ровно столько памяти, сколько нужно для маленького Info-файла, содержащего нужную информацию. Таким образом Emacs избегал лишнего расхода памяти.

Ссылки из одного файла в другой делались путем указания и имени файла, и имени ноды. (См. Раздел 7.5 [Ссылка на другие Info-файлы], с. 64. Также смотрите Раздел 8.3.4 [`@xref` с четырьмя и пятью аргументами], с. 70.)

Включение файлов было разработано преимущественно как способ создания единого, большого печатного руководства из нескольких меньших Info-файлов. В печатном руководстве все ссылки были внутри одного документа, так что TEX мог автоматически определить для них номера страниц. Команды форматирования Info использовали включаемые файлы лишь для создания объединенных именных указателей; каждый отдельный Texinfo-файл нужно было форматировать для Info индивидуально. (Каждый из них, следовательно, должен был содержать свою строку `@setfilename`.)

Однако, так как большие Info-файлы теперь автоматически разбиваются, больше нет необходимости сохранять их маленькими.

Теперь множественные Texinfo-файлы используются преимущественно для больших документов, таких как *The GNU Emacs Lisp Reference Manual* и для проектов, в которых несколько человек одновременно пишут разные разделы документа.

Кроме того, команды форматирования для Info были расширены для работы с командой `@include` так, чтобы создавать единый большой Info-файл, который разбивается при необходимости на меньшие файлы. Это означает, что вы можете писать меню и перекрестные ссылки не обращаясь к различным Texinfo-файлам.

# Приложение F Заголовки страниц

Большинство печатных руководств содержат заголовки вверху каждой страницы, исключая титульный лист и страницу с информацией об авторских правах. Некоторые руководства также содержат нижние заголовки. (Верхние и нижние заголовки не имеют значения для Info, в которой нет страниц.)

Texinfo предоставляет стандартные заголовки страниц для руководств, печатаемых на одной стороне каждого листа и для руководств, печатаемых на обоих сторонах листа. Как правило, вы будете использовать один из этих форматов, но вы также можете определить свой формат, если хотите.

Кроме того, вы можете указать, должны ли главы начинаться на новой странице или просто продолжать ту же страницу, где была предыдущая глава; и, если главы начинаются на новых страницах, вы можете указать, должны ли эти страницы быть нечетными.

По соглашению, книги печатаются на обоих сторонах каждого листа. Когда вы открываете книгу, правая страница имеет нечетный номер, и главы начинаются на правых страницах — предыдущая левая страница оставляется пустой, если необходимо. Отчеты, однако, часто печатаются только на одной стороне листа, и главы начинаются на новой странице, следующей сразу после конца предыдущей главы. В коротких или неофициальных отчетах главы часто вообще не начинаются на новой странице, а отделяются от предшествующего текста небольшим пропуском.

Команда `@setchapternewpage` контролирует, начинаются ли главы на новой странице, и какой из стандартных форматов заголовков используется. Помимо этого, в Texinfo есть несколько команд для верхних и нижних заголовков, которые вы можете использовать для создания ваших собственных форматов заголовков.

В Texinfo верхние и нижние заголовки — это одна строка в начале и одна в конце страницы; вы не можете создать многострочные заголовки. Каждый верхний или нижний заголовок разделен на три части: левую, среднюю и правую. Любая часть или вся строка может быть оставлена пустой. Текст левой части заголовка прижимается влево, текст средней части центрируется, а текст правой части прижимается вправо.

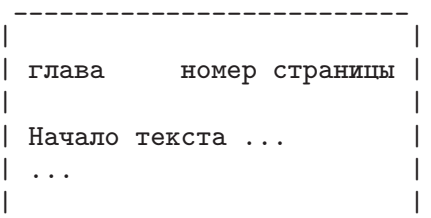
## F.1 Стандартные форматы заголовков

Texinfo предоставляет два стандартных формата заголовков: один для руководств, печатаемых на одной стороне листа, и второй для руководств, печатаемых на обоих сторонах листа.

По умолчанию нижний заголовок не задается, поэтому нижняя строка остается пустой.

Стандартный формат для печати на одной стороне листа состоит из строки заголовка, в которой левая часть содержит название главы, средняя часть пуста, а правая часть содержит номер страницы.

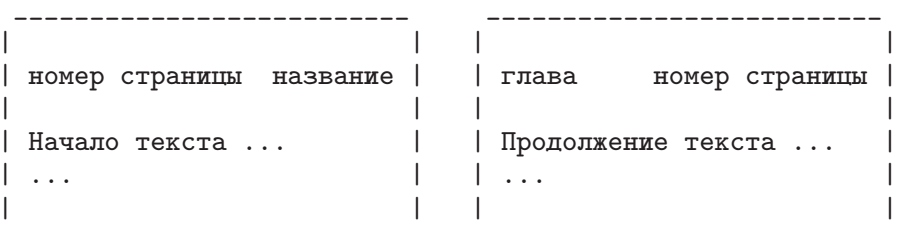
При печати на одной стороне листа страница выглядит так:




Стандартный формат для печати на двух сторонах зависит от того, четная это страница или нечетная. По соглашению, четные страницы находятся слева, а нечетные справа. (TEX сам настроит ширину правых и левых полей. Обычно поля получаются правильной ширины, но при двусторонней печати стоит проверить, что страницы соединятся верно — иногда принтер производит вывод, в котором четные страницу имеют более широкое правое поле, чем нечетные страницы.)

В стандартном формате для двусторонней печати левая часть левых (четных) страниц содержит номер страницы, средняя часть пуста, а правая часть содержит название книги (заданное командой `@settitle`). Левая часть правых (нечетных) страниц содержит название главы, средняя часть пуста, а правая часть содержит номер страницы.

Две последовательные страницы, как в раскрытой книге, выглядят так:




Перед названием главы печатается слово “Глава”, номер главы и двоеточие. Это помогает проследить, в каком месте руководства вы находитесь.

## F.2 Задание типа заголовка

TEX не начинает создавать заголовки страниц для стандартного Texinfo-файла, пока не достигнет команды `@end titlepage`. Таким образом, титульный лист и страница с информацией об авторских правах не нумеруются. Команда `@end titlepage` заставляет TEX начать создавать заголовки страниц в соответствии со стандартным форматом, заданным командой `@setchapternewpage`, идущей перед разделом `@titlepage`.

Возможны четыре варианта:

Команды `@setchapternewpage` нет
: Заставляет TEX использовать формат заголовков для односторонней печати и начинать главы на новой странице. Это аналогично применению `@setchapternewpage on`.

`@setchapternewpage on`
: Использовать формат заголовков для односторонней печати и начинать главы на новой странице.

`@setchapternewpage off`
: Заставляет TEX начинать главы на той же странице, где была предыдущая глава, сделав некоторый вертикальный пропуск. Также заставляет TEX производить набор для односторонней печати. (Вы можете переопределить формат заголовков с помощью команды `@headings double`; смотрите Раздел 3.4.6 [Команда `@headings`], с. 39.)

`@setchapternewpage odd`
: Использовать формат для двусторонней печати и начинать главы на новой странице.

В Texinfo нет команды `@setchapternewpage even`.

## F.3 Как создать свои заголовки

Вы можете использовать стандартные заголовки, предоставляемые Texinfo, или определить свои. По умолчанию, в Texinfo нет нижних заголовков, так что если вы определите их, доступный размер страницы для основного текста несколько уменьшится.

Texinfo предоставляет шесть команд для задания верхних и нижних заголовков:

- `@everyheading` и `@everyfooting` создают заголовки, одинаковые для четных и нечетных страниц.
- `@evenheading` и `@evenfooting` создают заголовки для четных (левых) страниц.
- `@oddheading` и `@oddfooting` для нечетных (правых) страниц.

Записывайте настройки заголовков в Texinfo-файле сразу после команды `@end titlepage`. Заключайте ваше описание между команд `@iftex` и `@end iftex`, так как команда `texinfo-format-buffer` может не распознать их. Также вы должны отменить предопределенные команды для заголовков командой `@headings off` до определения ваших собственных описаний.

Ниже показано, как сказать TEX помещать слева название главы, номер страницы посредине и дату справа в каждом заголовке как для четных, так и нечетных страниц:

```
@iftex
@headings off
@everyheading @thischapter @| @thispage @| @today{}
@end iftex
```

Вы должны отделять левую часть от средней и среднюю от правой, вставляя между частей ‘`@|`’. Иначе команда задания формата не сможет определить, где кончается текст одной части и начинается текст другой.

Каждая часть может содержать текст или `@`-команды. Текст печатается как если бы эта часть была обычным абзацем в теле страницы. `@`-команды заменяются на номер страницы, дату, название главы или что-либо еще.

Вот шесть команд для верхних и нижних заголовков:

`@everyheading` *левая* `@|` *средняя* `@|` *правая*
`@everyfooting` *левая* `@|` *средняя* `@|` *правая*
: Команды, начинающиеся с ‘every’, задают формат как для четных, так и для нечетных страниц. Эти команды предназначены для документов, печатаемых на одной стороне каждого листа бумаги или для таких, где вы хотите симметричные верхние или нижние заголовки.

`@evenheading` *левая* `@|` *средняя* `@|` *правая*
`@oddheading` *левая* `@|` *средняя* `@|` *правая*
`@evenfooting` *левая* `@|` *средняя* `@|` *правая*
`@oddfooting` *левая* `@|` *средняя* `@|` *правая*
: Команды, начинающиеся с ‘even’ и ‘odd’, задают формат для четных и нечетных страниц. Эти команды предназначены для книг и руководств, которые печатаются на обеих сторонах листа.

Используйте группу `@`-команд ‘`@this...`’ для получения названий глав и разделов и номеров страниц. Вы можете использовать команды ‘`@this...`’ в левой, средней или правой части и любом месте Texinfo-файла, если они заключены между команд `@iftex` и `@end iftex`.

Вот все команды ‘`@this...`’:

`@thispage`
: Раскрывается в номер текущей страницы.

`@thischaptername`
: Раскрывается в название текущей главы.

`@thischapter`
: Раскрывается в номер и название текущей главы, в формате ‘Глава 1: Название’.

`@thistitle`
: Раскрывается в название документа, указанное в команде `@settitle`.

`@thisfile`
: Только для включаемых файлов: раскрывается в имя текущего включаемого файла. Если текущий исходный Texinfo-файл не является включаемым, эта команда ничего не делает. Эта команда *не* предоставляет имя текущего исходного Texinfo-файла, если он не является включаемым. (См. Приложение E [Включаемые файлы], с. 202, для подробной информации о включаемых файлах.)

Вы также можете использовать команду `@today{}`, которая раскрывается в текущую дату, в формате ‘14 августа 1997г.’.

Другие `@`-команды и текст печатаются в заголовке точно так же, как бы они выводились в теле страницы. Удобно включать такой текст, особенно если вы пишете наброски:

```
@iftex
@headings off
@everyheading @emph{Набросок!} @| @thispage @| @thischapter
@everyfooting @| @| Версия: 0.27: @today{}
@end iftex
```

Будьте осторожны со слишком длинными названиями: они могут перекрыть другую часть заголовка и испортить его.

# Приложение G Ошибки форматирования

Помимо ошибок в содержимом документации, вы можете допустить в Texinfo еще два вида ошибок: ошибки в `@`-командах и ошибки в структуре нод и глав.

В Emacs есть два способа для нахождения ошибок в `@`-командах и два для ошибок в структуре.

Для нахождения ошибок в `@`-командах вы можете запустить TEX или команду форматирования области в той области, где возникла проблема; на самом деле вы можете запускать эти команды для каждой области по мере написания.

Для нахождения ошибок в структуре нод или глав, вы можете использовать команду *C-c C-s* (`texinfo-show-structure`) и другие команды пакета `occur`, а также команду *M-x Info-validate*.

Программа `makeinfo` проделывает отличную работу по нахождению ошибок и сообщению о них — намного лучшую, чем команда `texinfo-format-region` или `texinfo-format-buffer`. Кроме того, различные функции для автоматического создания и обновления указателей на ноды устраняют многие возможные ошибки человека.

Если есть возможность, применяйте команды обновления для создания и вставки указателей и меню. Они предотвращают появление многих ошибок. Потом используйте `makeinfo` (или ее проявления в режиме Texinfo, `makeinfo-region` и `makeinfo-buffer`) для форматирования вашего файла и проверки наличия других ошибок. Это наилучший способ работы с Texinfo. Однако, если вы не можете использовать `makeinfo`, или ваши ошибки слишком загадочны, вы можете применить инструменты, описанные в данном приложении.

## G.1 Поиск ошибок при форматировании для Info

После написания части Texinfo-файла, вы можете использовать команду `texinfo-format-region` или `makeinfo-region`, чтобы проверить, отформатируется ли область должным образом.

Но вероятнее всего вы читаете этот раздел, потому что по какой-либо причине не можете использовать команду `makeinfo-region`; поэтому в дальнейшем полагается, что вы пользуетесь командой `texinfo-format-region`.

Если вы допустили ошибку в `@`-команде, `texinfo-format-region` останавливается на этом месте или после него и выводит сообщение об ошибке. Чтобы увидеть, в каком месте буфера допущена ошибка, переключитесь в буфер ‘`*Info Region*`’; курсор будет находиться в позиции после места ошибки. Кроме того, текст после того места, где допущена (или, точнее, была обнаружена) ошибка, не будет отформатирован.

Например, если вы случайно завершите меню командой `@end menus` с ‘s’ в конце, вместо `@end menu`, вы увидите сообщение об ошибке, говорящее:

```
@end menus is not handled by texinfo
```

Курсор остановится в той точке буфера, где допущена ошибка или немного после. Буфер будет выглядеть так:

```
------- Buffer: *Info Region* -------
* Menu:

* Using texinfo-show-structure::  How to use
                                  'texinfo-show-structure'
                                  to catch mistakes.
* Running Info-Validate::         How to check for
                                  unreferenced nodes.
@end menus
*
------- Buffer: *Info Region* -------
```

Команда `texinfo-format-region` иногда выдает немного странные сообщения об ошибках. Например, следующая перекрестная ссылка не может быть отформатирована:

```
(@xref{Поиск ошибок, для подробной информации.)
```

В этом случае `texinfo-format-region` находит, что пропущена закрывающая фигурная скобка, но выводит сообщение, сообщающее о несбалансированных круглых, а не фигурных скобках. Это происходит, потому что форматирующая команда ищет непарные фигурные скобки, как если бы они были круглыми.

Иногда `texinfo-format-region` не может обнаружить ошибку. Например, во фрагменте ниже, закрывающая фигурная скобка поменялась местами с закрывающей круглой скобкой:

```
(@xref{Поиск ошибок), для подробной информации.}
```

Форматирование дает:

```
(*Note для подробной информации.: Поиск ошибок)
```

Единственный способ заметить эту ошибку — понять, что ссылка должна выглядеть так:

```
(*Note Поиск ошибок::, для подробной информации.)
```

Кстати, если вы читаете эту ноду в Info и введете *`f`* ⟨RET⟩ (`Info-follow-reference`), то получите такое сообщение об ошибке:

```
Невозможно найти ноду: "Поиск ошибок) Единственный ...
```

Это происходит, потому что Info воспринимает пример с ошибкой как первую перекрестную ссылку в этой ноде, и, если вы нажмете ⟨RET⟩ сразу после ввода команды Info *`f`*, попытается перейти к указанной ноде. Если вы введете *`f catch`* ⟨TAB⟩ ⟨RET⟩, Info завершит имя ноды правильно написанного примера и перенесет вас к ноде 'Поиск ошибок'. (Если вы попробуете так сделать, вы сможете вернуться из ноды 'Поиск ошибок', нажав *`l`* (`Info-last`).)

## G.2 Поиск ошибок при форматировании с TEX

Вы также можете найти ошибки при форматировании с помощью TEX.

Как правило, вам стоит сделать это после того, как вы прогнали `texinfo-format-buffer` (или лучше `makeinfo-buffer`) на том же файле, потому что иногда `texinfo-format-buffer` выводит более полезные сообщения об ошибках, чем TEX. (См. Раздел G.1 [Отладка для Info], с. 211, для подробной информации.)

Например, TEX был запущен для Texinfo-файла, часть которого приведена здесь:

```
------- Buffer: texinfo.texi -------
name of the Texinfo file as an extension.  The
@samp{??} are ‘wildcards’ that cause the shell to
substitute all the raw index files.  (@xref{sorting
indices, for more information about sorting
indices.)@refill
------- Buffer: texinfo.texi -------
```

(В перекрестной ссылке нет закрывающей фигурной скобки.) TEX выдает следующий вывод и останавливается:

```
------- Buffer: *tex-shell* -------
Runaway argument?
{sorting indices, for more information about sorting
indices.) @refill @ETC.
! Paragraph ended before @xref was complete.
<to be read again>
                   @par
l.27

?
------- Buffer: *tex-shell* -------
```

В данном случае TEX выдает точное и понятное сообщение об ошибке:

```
Абзац закончился до завершения @xref.
```

‘`@par`’ — это внутренняя команда TEX, не относящаяся к Texinfo. ‘`l.27`’ означает, что TEX обнаружил ошибку в строке 27 Texinfo-файла. ‘`?`’ служит приглашением для ввода, которое TEX использует в таких случаях.

К сожалению, сообщения TEX не всегда настолько полезны, и вам нужно быть по-настоящему Шерлоком Холмсом, чтобы понять, что за неприятность произошла.

В любом случае, если вы столкнулись с подобной проблемой, вы можете сделать одну из трех следующих вещей.

1. Вы можете сказать TEX продолжать выполнение и игнорировать эту ошибку, введя в ответ на приглашение ‘`?`’ символ ⟨RET⟩.
2. Вы можете сказать TEX продолжать выполнение и игнорировать все ошибки, насколько возможно, введя в ответ на приглашение ‘`?`’ символы *r* ⟨RET⟩.

   Часто это лучшее, что можно сделать. Однако, помните: одна ошибка может привести к каскаду дополнительных сообщений об ошибках, так как ее последствия сказываются до конца файла. Чтобы остановить TEX, когда он выводит такую лавину сообщений об ошибках, нажмите *C-c* (или *C-c C-c*, если вы запустили оболочку из Emacs).
3. Вы можете сказать TEX остановить работу, введя в ответ на приглашение ‘`?`’ символы *x* ⟨RET⟩.

Если вы запускаете TEX из Emacs, вы должны переключиться в буфер оболочки и перейти к строке, в которой TEX печатает ‘`?`’ в качестве приглашения для ввода.

Иногда TEX может отформатировать файл, не выдавая сообщений об ошибках, даже если ошибки есть. Обычно это случается, если команда не была завершена, но

TEX смог продолжить обработку. Например, если вы не завершите простой перечень командой `@end itemize`, TEX запишет DVI-файл, который вы сможете распечатать. Единственным сообщением об ошибке, которое выведет TEX, будет несколько загадочный комментарий

```
(@end occurred inside a group at level 1)
(команда @end встречена внутри группы на уровне 1)
```

Однако, если вы распечатаете DVI-файл, то обнаружите, что во всем тексте файла после перечня сделан отступ, как будто этот текст является частью последнего пункта перечня. Этим сообщением об ошибке TEX хотел сказать, что ожидал найти команду `@end` где-нибудь в файле; но он не может определить, где она действительно была нужна.

Другой источник пресловутых труднонаходимых ошибок — это пропущенная команда `@end group`. Если вы когда-нибудь окажетесь поставленными в тупик непонятными ошибками, поищите первым делом пропущенные команды `@end group`.

Если в Texinfo-файле пропущены строки заголовка, TEX может остановиться в начале работы и вывести что-то похожее на приведенное ниже. Символ '`*`' означает, что TEX ждет ввода.

```
This is TeX, Version 3.14159 (Web2c 7.0)
(test.texinfo [1])
*
```

В таком случае просто напечатайте `\end` ⟨RET⟩ после звездочки. Потом напишите в Texinfo-файле строки заголовка и запустите TEX снова. (Обратите внимание, вы должны использовать обратную косую черту, '\'. TEX использует '\' вместо '`@`'; а при этих обстоятельствах вы работаете непосредственно с TEX, а не с Texinfo.)

## G.3 Использование `texinfo-show-structure`

Не всегда легко уследить за нодами, главами, разделами и подразделами Texinfo-файла. В особенности это справедливо, если вы пересматриваете или дополняете Texinfo-файл, написанный кем-то другим.

В GNU Emacs в режиме Texinfo, команда `texinfo-show-structure` перечисляет все строки, начинающиеся `@`-командами, определяющими структуру: `@chapter`, `@section`, `@appendix` и так далее. Если задан аргумент (*C-u* в качестве числового аргумента, при интерактивном вызове), эта команда показывает также строки `@node`. В режиме Texinfo команда `texinfo-show-structure` по умолчанию привязана к ключу *C-c C-s*.

Строки выводятся в буфер, называемый буфером '`*Occur*`', с отступами по уровню иерархии. Например, вот часть полученного при запуске `texinfo-show-structure` в данном руководстве:

```
13 lines matching "^@\\(chapter \\|sect\\|subs\\|subh\\|
unnum\\|major\\|chapheading \\|heading \\|appendix\\)"
in buffer texinfo-06.texi.
   3:@chapter Ноды
  26:   @heading Два способа
  59:   @section Иллюстрация нод и меню
 167:   @section Команда @code{@@node}
 222:       @subheading Выбор имен нод и указателей
 245:       @subsection Как писать строку @code{@@node}
 294:       @subsection Советы по написанию строки @code{@@node}
 ...
```

Здесь написано, что строки 59, 222 и 245 файла ‘`texinfo-06.texi`’ начинаются командами `@section`, `@subheading` и `@subsection`, соответственно. Если вы переместите курсор в окно ‘`*Occur*`’, вы можете поместить курсор на одной из этих строк и перейти к соответствующему месту Texinfo-файла с помощью команды *C-c C-c* (`occur-mode-goto-occurrence`). См. раздел “Using Occur” в *Руководство по GNU Emacs*, для получения подробной информации о команде `occur-mode-goto-occurrence`.

Первая строка в окне ‘`*Occur*`’ описывает *регулярное выражение*, заданное в переменной *texinfo-heading-pattern*. Это регулярное выражение — тот образец, по которому производит поиск команда `texinfo-show-structure`. См. раздел “Using Regular Expressions” в *Руководство по GNU Emacs*, для получения подробной информации.

Когда вы вызываете команду `texinfo-show-structure`, Emacs отобразит структуру всего буфера. Если вы хотите посмотреть структуру только части буфера, например одной главы, используйте команду *C-x n n* (`narrow-to-region`) для пометки области. (См. раздел “Narrowing” в *Руководство по GNU Emacs*.) Так был получен пример выше[1]. (Чтобы снова увидеть весь буфер, используйте *C-x n w* (`widen`).)

Если вы вызовете `texinfo-show-structure` с числовым аргументом, напечатав *C-u C-c C-s*, будут перечислены строки, начинающиеся с `@node`, а так же строки, начинающиеся `@`-командами `@chapter`, `@section` и подобными.

Вы можете вспомнить структуру Texinfo-файла, взглянув на список в окне ‘`*Occur*`’; и если вы неправильно назвали ноду или пропустили раздел, то сможете исправить ошибку.

## G.4 Использование `occur`

Иногда команда `texinfo-show-structure` выдает слишком много информации. Вероятно, вы хотите вспомнить общую структуру Texinfo-файла, и вам не нужен слишком детальный список, выдаваемый командой `texinfo-show-structure`. в таком случае вы можете непосредственно использовать команду `occur`. Для этого напечатайте

*M-x occur*

и затем в ответ на приглашение напечатайте *регулярное выражение*, образец для поиска. (См. раздел “Regular Expressions” в *Руководство по GNU Emacs*.) Команда

[1] При переводе руководство было разбито на несколько файлов по главам, поэтому не было необходимости использовать этот метод. *(Прим. переводчика)*

`occur` начинает работу от текущей позиции курсора до конца буфера. Если вы хотите запустить `occur` для буфера целиком, поместите курсор в начале буфера.

Например, чтобы увидеть все строки, содержащие слово '`@chapter`', просто напечатайте '`@chapter`'. Вы получите список всех глав. Также будут перечислены все предложения со словом '`@chapter`' в середине строки.

Если вы хотите увидеть только строки, начинающиеся словом '`@chapter`', напечатайте на запрос `occur` '`^@chapter`'. Если вы хотите увидеть все строки, заканчивающиеся определенным словом или фразой, завершите последнее слово символом '`$`', например '`поиск ошибок$`'. Это может быть полезно, если вы хотите увидеть все ноды, являющиеся частью одной и той же главы или раздела, и, следовательно, имеющие один и тот же указатель 'Up'.

См. раздел "Using Occur" в *Руководство по GNU Emacs*, для большей информации.

## G.5 Поиск неправильных ссылок на ноды

Вы можете использовать команду `Info-validate` для проверки, ссылаются ли 'Next', 'Previous', 'Up' или другие указатели на ноды. Эта команда проверяет, что каждый указатель ссылается на существующую ноду. Команда `Info-validate` работает только с Info-файлами, но не с Texinfo-файлами.

Программа `makeinfo` проверяет указатели автоматически, так что вам не придется применять команду `Info-validate`, если вы используете `makeinfo`. Вам понадобится применять `Info-validate`, если вы не можете запустить `makeinfo` и вместо этого должны создавать Info-файл, используя `texinfo-format-region` или `texinfo-format-buffer`, или если вы пишите Info-файл с нуля.

### G.5.1 Запуск `Info-validate`

Чтобы использовать `Info-validate`, обратитесь к Info-файлу, который вы хотите проверить и напечатайте:

```
M-x Info-validate
```

Обратите внимание, в команде `Info-validate` должна стоять заглавная буква 'I'. Вы также должны создать таблицу тегов перед запуском `Info-validate`. См. Раздел G.5.3 [Теги], с. 217.

Если файл написан правильно, вы получите сообщение "File appears valid". Однако, если есть указатель, ссылающийся на несуществующую ноду, в буфер '`*problems in info file*`' будет выведено сообщение об ошибке.

Например, `Info-validate` была запущена в тестовом файле, содержащем только первую ноду данного руководства. Одно из сообщений говорит:

```
In node "Обзор", invalid Next: Режим Texinfo
```

Это значит, что нода, называемая '`Обзор`', содержит указатель 'Next', который ни на что не ссылается (что верно в данном случае, так как в этом тестовом файле есть только одна нода).

Теперь предположим, что мы добавили ноду, называемую '`Режим Texinfo`', к нашему тестовому файлу, но не задали для нее указатель 'Previous'. Тогда мы получим следующее сообщение об ошибке:

```
In node "Режим Texinfo", should have Previous: Обзор
```

Это происходит, потому что каждому указателю ‘Next’ должен соответствовать указатель ‘Previous’ (в ноде, на которую указывает ‘Next’), ссылающийся назад.

`Info-validate` также проверяет, все ли пункты меню и перекрестные ссылки ссылаются на действительные ноды.

`Info-validate` требует наличия таблицы тегов и не работает с разбитыми файлами. (Команда `texinfo-format-buffer` автоматически разбивает большие файлы.) Чтобы использовать `Info-validate` с большим файлом, вы должны запустить `texinfo-format-buffer` с аргументом, чтобы она не разбивала Info-файл; и вы должны создать для этого неразбитого файла таблицу тегов.

### G.5.2 Создание неразбитого файла

Вы можете запустить `Info-validate` только для одного Info-файла, имеющего таблицу тегов. Эта команда не работает с косвенными подфайлами, созданными при разбиении главного файла. Если у вас есть большой файл (длиннее 70000 байт или около этого), вам нужно запускать команды `texinfo-format-buffer` или `makeinfo-buffer` таким образом, чтобы они не создавали косвенных подфайлов. Вам также нужно будет создать для Info-файла таблицу тегов. После того, как вы это сделали, вы можете запускать `Info-validate` и искать неправильные ссылки на ноды.

Первый шаг — создание неразбитого Info-файла. Чтобы запретить `texinfo-format-buffer` разбивать Texinfo-файл на меньшие Info-файлы, задайте команде *`M-x texinfo-format-buffer`* аргумент:

```
C-u M-x texinfo-format-buffer
```

или, иначе

```
C-u C-c C-e C-b
```

Если вы сделаете так, Texinfo не будет разбивать файл и создавать для него таблицу тегов.

### G.5.3 Создание тегов в файле

После создания неразбитого Info-файла вы должны создать для него таблицу тегов. Обратитесь к Info-файлу, в котором вы хотите сделать таблицу тегов и напечатайте:

```
M-x Info-tagify
```

(Обратите внимание на заглавную букву ‘`I`’ в `Info-tagify`.) Это создаст Info-файл с таблицей тегов, который вы сможете проверить.

Третий шаг — проверить Info-файл:

```
M-x Info-validate
```

(Обратите внимание на заглавную букву ‘`I`’ в `Info-validate`.) Кратко, шаги таковы:

```
C-u M-x texinfo-format-buffer
M-x Info-tagify
M-x Info-validate
```

После проверки структуры нод вы можете перезапустить `texinfo-format-buffer` обычным способом, так что она автоматически сделает таблицу тегов и разобьет файл, или вы можете создать таблицу тегов и разбить файл вручную.

### G.5.4 Разбивание файла вручную

Вам стоит разбивать большие файлы или предоставлять командам `texinfo-format-buffer` или `makeinfo-buffer` делать это для вас автоматически. (Обычно вы будете предоставлять эту работу одной из команд форматирования. См. Раздел 20.1 [Создание Info-файла], с. 158.)

Файлы, получаемые в результате разбиения, называются косвенными подфайлами.

Info-файлы разбиваются для экономии памяти. Работая с меньшими файлами, Emacs не должен создавать большие буферы для хранения информации.

Если в Info-файле более 30 нод, вам стоит также создать для него таблицу тегов. См. Раздел G.5.1 [Использование Info-validate], с. 216, для получения информации о методе создания таблицы тегов. (Опять же, таблицы тегов обычно создаются автоматически командой форматирования; вам нужно создавать таблицу тегов самим, только если вы пишите файл вручную. Скорее всего, вам придется это делать для большого неразбитого файла, в котором вы хотите запустить `Info-validate`.)

Обратитесь к Info-файлу, который вы хотите разбить и снабдить таблицей тегов, и напечатайте две команды:

```
M-x Info-tagify
M-x Info-split
```

(Обратите внимание, буква ‘`I`’ в ‘`Info`’ — заглавная.)

Когда вы примените команду `Info-split`, буфер сменится на (маленький) Info-файл, в котором перечислены косвенные подфайлы. Этот файл должен быть записан вместо файла, к которому вы первоначально обратились. Косвенные подфайлы записываются в тот же каталог, где находился первоначальный файл, а их имена генерируются добавлением к имени первоначального файла символа ‘`-`’ и числа.

Главный файл продолжает работать в качестве Info-файла, но содержит лишь таблицу тегов и каталог подфайлов.

# Приложение H Перезаполнение абзацев

Команда `@refill` заполняет абзац и, возможно, делает отступ в его первой строке.[1] Команда `@refill` уже не важна, но мы описываем ее, так как она была нужна раньше. Вы увидите ее во многих старых Texinfo-файлах.

Без перезаполнения, абзацы, содержащие длинные `@`-конструкции могли выглядеть после форматирования плохо, потому что программа форматирования удаляет `@`-команды и укорачивает некоторые строки больше других. Раньше ни команда `texinfo-format-region`, ни `texinfo-format-buffer` не перезаполняли абзацы автоматически. Нужно было писать команду `@refill` в конце каждого абзаца, чтобы эти форматирующие команды заполняли их. (И TEX и `makeinfo` всегда перезаполняли абзацы автоматически.) Теперь все программы, форматирующие для Info, автоматически заполняют и делают отступы в абзацах, которые этого требуют.

Команда `@refill` заставляет `texinfo-format-region` и `texinfo-format-buffer` перезаполнять абзац в Info-файле *после* того, как завершена вся другая обработка. По этой причине вы не можете использовать `@refill` в абзаце, содержащем `@*` или `@w{ ... }`, так как перезаполнение перекроет эти команды.

Команды `texinfo-format-region` и `texinfo-format-buffer` теперь автоматически добавляют `@refill` в конец каждого абзаца, который должен быть заполнен. Они не добавляют `@refill` в конец абзацев, которые содержат `@*` или `@w{ ...}`, и, следовательно, не перезаполняют их и не делают в них отступы.

[1] Может быть, эту команду стоило назвать `@refillandindent`, но `@refill` короче, и имя было выбрано до того, как стали возможны отступы.

# Приложение I Синтаксис @-команд

Символ '@' используется для начала специальных команд Texinfo. (Он имеет то же значение, что и '\' в plain TEX.) В Texinfo есть четыре вида @-команд:

1. Неалфавитные команды.
   Эти команды состоят из @, за которым идет знак препинания или другой символ, не входящий в алфавит. Неалфавитные команды почти всегда являются частью текста внутри абзаца и никогда не принимают аргументов. Два символа (@ и следующий) самодостаточны; ни за одной из этих пар не пишутся фигурные скобки. Вот все неалфавитные команды: `@.`, `@:`, `@*`, `@SPACE`, `@TAB`, `@NL`, `@@`, `@{` и `@}`.

2. Алфавитные команды, не требующие аргументов.
   Эти команды начинаются с @, за которым идут левая и правая фигурные скобки. Эти команды вставляют в документ специальные символы; они не требуют аргументов. Например, `@dots{}` ⇒ '...', `@equiv{}` ⇒ '≡', `@TeX{}` ⇒ 'TEX' и `@bullet{}` ⇒ '•'.

3. Алфавитные команды, принимающие аргументы в фигурных скобках.
   Эти команды начинаются с @, за которым идут буква или слово и аргумент в фигурных скобках. Например, команда `@dfn` обозначает вводимый или определяемый термин; она используется следующим образом: '`В Texinfo, @@-команды -- это команды @dfn{разметки}.`'

4. Алфавитные команды, занимающие целую строку.
   Эти команды занимают всю строку. Строка начинается с @, за которым идет имя команды (слово); например, `@center` или `@cindex`. Если не требуется аргумент, после слова пишется символ конца строки. Если нужен аргумент, он отделяется от имени команды пробелом. Фигурные скобки не используются.

Таким образом, алфавитные команды разделяются на классы с разными синтаксисами аргументов. Нельзя сказать, к какому классу принадлежит команда по виду ее имени, но можно определить это по ее значению: если команда обозначает графический знак, то она относится к классу 2 и не требует аргументов; если важно использовать команду вместе с другим текстом в абзаце, то она относится к классу 3, и за ней должен следовать аргумент в фигурных скобках; иначе она относится к классу 4 и использует остаток строки в качестве аргумента.

Команды классов 3 и 4 имеют разный синтаксис, чтобы было легче читать Texinfo-файлы, а так же чтобы помочь правильной работе команд GNU Emacs для заполнения и работы с абзацами. Есть только одно исключение из этого правила: команда `@refill`, которая всегда используется в конце абзаца, сразу после завершающей точки или другого знака препинания. `@refill` не принимает аргументов и *не* требует фигурных скобок. `@refill` никогда не смущает команды Emacs для абзацев, потому что никогда не появляется в начале строки.

# Приложение J Как получить TEX

TEX распространяется свободно. Вы можете получить TEX для Unix-систем по анонимному ftp или на физическом носителе. Основной материал состоит из пакета Web2c TEX (http://tug.org/web2c).

Инструкции по загрузке по анонимному ftp и сведения о других доступных дистрибутивах:

ftp://tug.org/tex/unixtex.ftp
http://tug.org/unixtex.ftp

Фонд свободного программного обеспечения предоставляет основной дистрибутив, достаточный для печати руководств на Texinfo, на *Source Code CD-ROM*. Вы можете заказать его по этому адресу:

Free Software Foundation, Inc.
59 Temple Place Suite 330
Boston, MA 02111-1307
USA
Телефон: +1-617-542-5942
Факс: (включая Японию) +1-617-542-2652
Бесплатный факс (в Японии):
0031-13-2473 (KDD)
0066-3382-0158 (IDC)
Электронная почта: gnu@gnu.org

Существует много других дистрибутивов TEX; смотрите http://tug.org/.

# Указатель команд и переменных

Это алфавитный список всех @-команд, а также функций Emacs Lisp и некоторых переменных. Чтобы списком было удобнее пользоваться, команды перечислены без начального символа '@'.

## default



## A



## B



## C



## D

## E



## F



## G



## H



## I



## K



## L

## M



## N



## O



## P



## Q



## R



## S

## T



## U



## V



## W



## X

# Указатель понятий

## default



## A



## B



## C



## D



## E

## F



## G



## H



## I



## J



## L



## M



## N



## O



## P



## R



## S

## T



## U



## A



## Б



## B

## Г



## Д



## Е



## З



## И

## К



## Л



## М



## Н

## О



## П

## P

## C

## Т



## У



## Ф



## Х



## Ц



## Ч

## Ш



## Э



## Я