Создание цветных иллюстрированных документов на PostScript и в PDF с помощью LaTeX'а

© Владимир Сюткин

Новосибирск, syutkin@ns.kinetics.nsc.ru*

21 октября 2001 г.

Аннотация

С помощью пакетов из коллекции graphics (автор David Carlisle) можно включать в документ рисунки из графических файлов, поворачивать, растягивать или сжимать произвольным образом любой блок на странице, выбирать по своему усмотрению цвет текста и фона как отдельного блока, так и всей страницы. Раскрасить таблицы нам поможет пакет colortbl (автор David Carlisle). В статье описаны практически все возможности, которые дают нам пакеты color, graphicx и lscape из коллекции graphics и пакет colortbl при создании документов, описанных на PostScript'е и в формате PDF в fpTeX-0.4. В конце статьи кратко описано, какие навигационные элементы в документе PDF создаёт пакет hyperref.

Содержание

*Замечания, пожелания и предложения по содержанию документа приветствуются.

1 Выбор драйвера

При загрузки пакетов из коллекции graphics надо в необязательном аргументе команды `\usepackage` указать драйвер, который будет переводить свёрстанный TEX'ом документ в формат, приемлемый как для просмотра документа на экране монитора, так и для получения бумажной копии высокого качества. В fpTEX'е такими драйверами являются `dvips` и `pdftex`. Они используют векторные шрифты Type 1 и поддерживают включение в документ рисунков из файлов в форматах EPS (Encapsulated PostScript) и PDF (Portable Document Format), соответственно. Драйвером по умолчанию задан `dvips`[1]. Программа `dvips` (автор Tomas Rokicki) переводит `dvi`-файл, полученный после обработки LATEX'ом входного файла, в файл,

[1] Драйвер по умолчанию задаётся в аргументе команды `\ExecuteOptions` в файлах настройки `graphics.cfg` и `color.cfg`.

в котором документ описан на языке PostScript. `ps`-файл можно с помощью программы `GhostScript`[2] просмотреть на экране монитора и распечатать на обычном принтере. Кроме того, `GhostScript` конвертирует `ps`-файл в файл формата PDF. Если же вы создаёте документ в формате PDF непосредственно с помощью pdfLaTeX'а, то вам следует при загрузки пакетов из коллекции graphics указывать в виде опции драйвер `pdftex`. Поскольку опция с именем драйвера является общей для всех пакетов коллекции, её можно указать прямо в команде `\documentclass`.

На заметку Один и тот же входной файл можно обрабатывать как LaTeX'ом, так и pdfLaTeX'ом, если использовать во входном файле конструкцию из примера

```
\ifx\pdfoutput\undefined
\documentclass{article}
\else
\documentclass[pdftex]{article}
\fi
```

2 Оформление цветных документов

Пакет color позволяет нам выбирать по своему усмотрению цвет текста и фона как отдельного блока на странице, так и всей страницы печатного документа.

2.1 Как задать цвет

Цвет, в общем случае, задаётся двумя параметрами: цветовой моделью *model* (задаётся в командах в виде опции) и спецификацией *spec* (задаётся как обязательный аргумент команд). В этом разделе описаны цветовые модели пакета color и синтаксис спецификации цвета для каждой из них. В примерах используются команды, синтаксис которых описан в разделе 2.2.

Цветовая модель `named`

`named` является цветовой моделью по умолчанию. Спецификация цвета в этой модели задаётся по имени цвета. В пакете color определены имена восьми цветов, приведённые в следующей таблице вместе с самими цветами:

Имя		Цвет	Имя		Цвет
`black`	чёрный		`white`	белый	
`red`	красный		`green`	зелёный	
`blue`	синий		`cyan`	голубой	
`magenta`	пурпурный		`yellow`	жёлтый	

[2] `GhostScript` является интерпретатором языка PostScript и используется в среде Windows под оболочкой `GSview`.

Пример с командой `\textcolor` из раздела 2.2[3]:

```
\textcolor{blue}{Синий $x^2$ и \dots}
```

Синий x^2 и ...

Видно, что команды переключения цвета текста изменяют цвет не только в текстовой моде, но и в математической.

На заметку Цвета, которые вы видите сейчас на этой странице, зависят от выходного устройства, на котором вы смотрите документ. Даже на экран монитора GhostScript и Acrobat Reader выводят разные цвета с одной и той же спецификацией. Что уж тут говорить о дешёвых принтерах.

Драйвер `dvips` содержит в файле `dvipsnam.def` определение имён ещё 68 цветов. Они заданы в цветовой модели `cmyk`, описанной ниже.

Имя	Цвет	Имя	Цвет
GreenYellow		Yellow	
Goldenrod		Dandelion	
Apricot		Peach	
Melon		YellowOrange	
Orange		BurntOrange	
Bittersweet		RedOrange	
Mahogany		Maroon	
BrickRed		Red	
OrangeRed		RubineRed	
WildStrawberry		Salmon	
CarnationPink		Magenta	
VioletRed		Rhodamine	
Mulberry		RedViolet	
Fuchsia		Lavender	
Thistle		Orchid	
DarkOrchid		Purple	
Plum		Violet	
RoyalPurple		BlueViolet	
Periwinkle		CadetBlue	
CornflowerBlue		MidnightBlue	
NavyBlue		RoyalBlue	
Blue		Cerulean	
Cyan		ProcessBlue	
SkyBlue		Turquoise	
TealBlue		Aquamarine	
BlueGreen		Emerald	
JungleGreen		SeaGreen	
Green		ForestGreen	
PineGreen		LimeGreen	

продолжение на следующей странице

[3]Здесь и далее в левой части примера показан исходный текст, а в правой — результат его обработки LaTeX'ом.

(продолжение)			
Имя	Цвет	Имя	Цвет
YellowGreen		SpringGreen	
OliveGreen		RawSienna	
Sepia		Brown	
Tan		Gray	
Black		White	

Если пакет color загружен с опцией `usenames`:

```
\usepackage[usenames]{color}
```

то цвета драйвера `dvips` можно использовать точно так же, как и цвета пакета color, указывая их имена в качестве спецификации цвета. Пример:

```
\textcolor{Orange}{Оранжевый $\bigcup$.}
```

Оранжевый $\bigcup$.

Если опция `usenames` опущена, то при использовании цветов драйвера `dvips` надо указывать опцию `named` в качестве цветовой модели:

```
\color[named]{Orange} Оранжевый $\oint$.
```

Оранжевый $\oint$.

На заметку Имена цветов драйвера `dvips` распознаются драйвером `pdftex`, если пакет color загружен с опцией `dvipsnames`.

Цветовая модель `rgb`

В модели `rgb` любой цвет получается в результате смешения лучей света трёх базовых цветов: красного (red), зелёного (green) и синего (blue). Поэтому спецификация цвета в этой модели задаётся тремя перечисленными через запятую числами от 0 до 1, которые соответствуют абсолютной интенсивности базовых составляющих света. Например, жёлтый цвет задаётся как `1,1,0`, а белый — как `1,1,1`. `0,0,0` даёт чёрный цвет.

Задавая цвет, надо помнить, что его восприятие зависит не только от длины волны света, но и от насыщенности и яркости излучения. Например, свет, содержащий лучи только красного цвета, от тусклого источника воспринимается не как бледно красный, а как чёрный. Бледно красный цвет получается при большой интенсивности света, в котором красная составляющая несколько превышает остальные.

Цветовая модель `gray`

Серый (gray) цвет получается при смешении в равных пропорциях базовых цветов из модели `rgb` с интенсивностью, меньшей единицы. Спецификация цвета в модели `gray` задаётся одним числом от 0 до 1, например, `0.5` вместо `0.5,0.5,0.5` в модели `rgb`. Чёрному цвету соответствует `0`, а белому — `1`. Пример со спецификацией `gray` можно найти в разделе 2.1.1.

Цветовая модель `cmyk`

Если на лист бумаги нанести красную краску и затем осветить его белым светом, то только красная составляющая света отразится от бумаги. Нанесём теперь на бумагу красную краску густо посаженными точками, а затем заполним все промежутки между красными точками зелёной краской. Теперь от листа бумаги отразится как красный, так и зелёный свет. Издали различить отдельные точки нельзя, поэтому глаз получит смесь красного и зелёного света и бумага будет выглядеть жёлтой[4]. На этом принципе основана цветовая модель `cmyk`. В этой модели спецификация цвета задаётся четырьмя перечисленными через запятую числами от 0 до 1, которые соответствуют «количеству» голубой (cyan), пурпурной (magenta), жёлтой (yellow) и чёрной (black) краски на белом листе бумаги. Теоретически при смешении первых трёх цветов в равной пропорции должен получиться чёрный цвет. В действительности краски поглощают свет не полностью и поэтому смесь трёх основных цветов выглядит тёмно-коричневой. По этой причине в модель введена ещё и чёрная краска.

Из описанного выше ясно, что в модели `cmyk` белому цвету соответствует спецификация `0,0,0,0` (на белом листе бумаги нет никакой краски), а чёрному — `0,0,0,1`.

2.1.1 Определение имени цвета

Команда

`\definecolor{`*name*`}{`*model*`}{`*spec*`}`

позволяет определить имя *name* для любого цвета. Здесь *model* — цветовая модель, *spec* — спецификация цвета. Используя *name* в качестве спецификации цвета, цветовую модель можно не указывать, поскольку они относятся к модели по умолчанию `named`. Пример:

```
\definecolor{faded}{gray}{0.7}
\textcolor{faded}{Блёклый\dots}
```

Блёклый. . .

[4] Если сначала смешать красную и зелёную краски, а затем нанести смесь на бумагу, то получится тёмный цвет с красноватым оттенком.

2.2 Цветной текст

Изменить цвет текста *text* в документе можно либо командой

`\textcolor[`*model*`]{`*spec*`}{`*text*`}`

либо декларацией

`{\color[`*model*`]{`*spec*`}`*text*`}`

Здесь *model* — цветовая модель, *spec* — спецификация цвета. Пример с разными цветовыми моделями:

```
\textcolor{red}{Красный,}
\textcolor[cmyk]{0,1,1,0}{опять красный,}
\textcolor[rgb]{1,0,0}{и ещё раз красный!}
```

Красный,
опять красный,
и ещё раз красный!

На заметку Команды включения цвета из пакета color игнорируются, если пакет загружен с опцией `monochrome`. Используется, когда выходное устройство не поддерживает цвета.

2.3 Цветной фон блока

Команда

`\colorbox[`*model*`]{`*spec*`}{`*lr-text*`}`

помещает, подобно команде `\mbox`, текст *lr-text* в бокс, у которого цвет фона задан аргументами *model* и *spec*. Здесь *model* — цветовая модель, *spec* — спецификация цвета. Команда

`\fcolorbox[`*model*`]{`*fr-spec*`}{`*spec*`}{`*lr-text*`}`

дополнительно обводит этот бокс рамкой цвета *fr-spec*. Пример:

```
\fcolorbox{red}{yellow}{Текст\dots}
```

Текст. . .

В этом примере опция, задающая цветовую модель опущена, поскольку мы используем цвета из пакета color. Список таких цветов приведён в разделе 2.1.

На заметку В качестве толщины линий рамки и ширины промежутка между рамкой и текстом в боксе команда `\fcolorbox` использует значения параметров `\fboxrule` и `\fboxsep`.

2.4 Цветной фон страницы

Изменить цвет страницы можно декларацией

`\pagecolor[`*model*`]{`*spec*`}`

Здесь *model* — цветовая модель, *spec* — спецификация цвета. Область действия декларации не ограничивается никакими скобками. Чтобы вернуть белый цвет страниц, надо вызвать команду `\pagecolor{white}`.

2.5 Цветные таблицы

Пакет colortbl (автор David Carlisle) совместно с пакетами color и array позволяет нам раскрашивать таблицы. Цвета задаются точно так же, как в пакете color через цветовую модель *model* и спецификацию цвета *spec* (см. раздел 2.1).

Команду

`\columncolor[`*model*`]{`*spec*`}[`*left-overhang*`][`*right-overhang*`]`

можно использовать в спецификации `>{...}` (её вводит пакет array) в преамбуле окружений `tabular` и `array`. Она окрашивает весь столбец таблицы цветом, который задаётся первыми двумя аргументами *model* и *spec*. Опции *left-overhang* и *right-overhang* задают расстояния, соответственно, слева и справа между краями окрашенной области и текстом. Если указан только один аргумент, то он задаёт оба расстояния. По умолчанию значения аргументов задаются командными длинами `\tabcolsep` для окружения `tabular` и `\arraycolsep` для окружения `array`. Следующий пример показывает, как оставить в средней колонке неокрашенными полоски шириной `.4\tabcolsep`:

```
\begin{tabular}{%
|>{\color{white}\columncolor{black}}l|
 >{\columncolor{yellow}[.6\tabcolsep]}c|
 >{\columncolor[gray]{.8}}r|}
один & два & три \\ четыре & пять & шесть
\end{tabular}
```

один	два	три
четыре	пять	шесть

Команда

`\rowcolor[`*model*`]{`*spec*`}[`*left-overhang*`][`*right-overhang*`]`

используется для окрашивания целой строки таблицы. Её место — в самом начале строки:

```
\begin{tabular}{|l|c|}
\rowcolor[gray]{.9} один & два \\
\rowcolor[gray]{.6} три & четыре
\end{tabular}
```

один	два
три	четыре

Приведём теперь пример таблицы со слитыми ячейками в строке. В этом случае цвет слитых ячеек нужно задавать в спецификации команды `\multicolumn`. Я сделал это, введя новый тип колонки:

```
\newcolumntype{H}{>{\columncolor{magenta}}c}
\begin{tabular}{%
|>{\columncolor{yellow}}l|
>{\color{white}\columncolor{black}}l|}
\multicolumn{2}{|H|}{один} \\
 два & три \\ четыре & пять
\end{tabular}
```

один	
два	три
четыре	пять

На заметку Пакет colortbl совместим с пакетами longtable и dcolumn.

Цветные таблицы лучше смотрятся, когда строки в них отделены друг от другой белым промежутком. Для вставки такого промежутка можно определить новую команду. Назовём её `\tabrowsep`. Использую более компактный синтаксис plain TEX'a, определим команду `\tabrowsep` в виде:

```
\def\tabrowsep{\noalign{\vskip 2pt}}
```

Здесь используется команда `\noalign`, которая вставляет бокс (в нашем случае высотой 2 pt) в стопку боксов (в нашем случае это будут строки таблицы). Приведём пример с командой `\tabrowsep`:

```
\newcolumntype{H}{%
>{\columncolor[gray]{.9}}p{1.7cm}}
\begin{tabular}[t]{*2H}
\rowcolor[gray]{.6}один & два\\\tabrowsep
три & четыре \\\tabrowsep пять & шесть
\end{tabular}
```

один	два
три	четыре
пять	шесть

Глобальная декларация

`\arrayrulecolor[`*model*`]{`*spec*`}`

задаёт цвет горизонтальных и вертикальных линий, разделяющих ячейки в таблицах. Её можно вводить не только перед таблицей, но и в спецификации `>{...}` в преамбуле таблицы или в начале какой-нибудь строки таблицы. Область действия декларации `\arrayrulecolor` начинается в точке её размещения во входном файле. Так, если ввести декларацию сразу после преамбулы, то вертикальные разделительные линии, заданные в преамбуле, сохранят свой цвет.

Ещё одна глобальная декларация

`\doublerulesepcolor[`*model*`]{`*spec*`}`

задаёт окрашивание промежутка между двойными вертикальными (`||`) и двойными горизонтальными (`\hline\hline`) разделительными линиями в таблицах.

Ширину линий и промежутка между линиями можно выбрать по своему усмотрению. Пример с обеими декларациями:

```
\setlength\arrayrulewidth{2pt}\arrayrulecolor{blue}
\setlength\doublerulesep{2pt}\doublerulesepcolor{yellow}

\begin{tabular}{||l||c||}
\hline\hline
один & два \\ три & четыре \\
\hline\hline
\end{tabular}
```

Совет Сохраните начальное значение ширины линейки и тогда вам, если потребуется вернуться к первоначальной ширине, не придётся вспоминать его:

```
\newlength{\ArrayrulewidthDefault}
\setlength{\ArrayrulewidthDefault}{\arrayrulewidth}
...
\setlength{\arrayrulewidth}{\ArrayrulewidthDefault}
```

Команда `\hhline` из пакета hhline, рисующая горизонтальные линии в таблицах, позволяет модифицировать таблицу из предыдущего примера, например, так:

```
\begin{tabular}{||l|c||}
\hhline{|t:=:t:=:t|}
один & два \\ три & четыре \\
\hhline{|b:=:b:=:b|}
\end{tabular}
```

Декларации `\arrayrulecolor` и `\doublerulesepcolor` можно использовать в `>{...}` перед `-` и `=` в аргументе команды `\hhline`. Пример:

```
\newcommand\rgbline[1]{\hhline{>{\arrayrulecolor{red}}|#1:=%
>{\arrayrulecolor{green}}=>{\arrayrulecolor{blue}}=:#1|}}

\arrayrulecolor{red}
\begin{tabular}{|||*3c||}\rgbline{t}
red & green & \multicolumn{1}{c||}{blue}\\
\rgbline{b}\end{tabular}
```

На заметку Из-за проблем с окраской линий `\cline`, используйте вместо неё команду `\hhline` с линейкой `-`.

3 Выбор формата рисунка

3.1 Драйвер `dvips`

Драйвер `dvips` поддерживает включение в документ рисунков только из файлов в формате EPS[5]. В `eps`-файле рисунок описан командами PostScript'а. Кроме того, он может содержать растровые картинки. `eps`-файл имеет в обязательном порядке строку:

`%%BoundingBox:` *llx lly urx ury*

где целые числа *llx*, *lly*, *urx* и *ury* — это x- и y-координаты в больших пунктах[6] левого нижнего и правого верхнего углов области, внутри которой находится рисунок на «воображаемой» странице. Именно эта часть страницы импортируется в документ.

В наших примерах будет использоваться файл `a.eps`, в котором область с рисунком задана как:

```
%%BoundingBox: 99 99 155 155
```

Следовательно, точка отсчёта рисунка (левый нижний угол) находится на расстоянии 99 bp как от левого, так и от нижнего краёв страницы, а сам рисунок имеет размер 56 bp на 56 bp или примерно 1.98 см на 1.98 см.

Если необходимо вставить в документ растровый рисунок, то надо сначала преобразовать файл с этим рисунком в `eps`-файл. Многие графические редакторы и программы сохраняют растровую графику в виде `eps`-файла. Мы обсудим здесь только две программы, которые входят в fpTEX и делают эту работу гораздо лучше любой программы, включая даже GhostScript.

Программа `jpeg2ps.exe` преобразует файл в формате JPEG в `eps`-файл. При этом сам рисунок не конвертируется в формат EPS. `jpeg2ps` записывает в заголовок `eps`-файла информацию о параметрах рисунка, а затем копирует туда JPEG-данные. Распаковка данных осуществляется интерпретатором PostScript'а, например, программой `GhostScript`, на этапе просмотра или печати рисунка. Для преобразования `jpg`-файла в `eps`-файл достаточно выполнить следующую командную строку:

`jpeg2ps -r 0 -o` *outputfile*`.eps` *inputfile*`.jpg`

Опция `-o` переводит программу `jpeg2ps` в режим записи выходных данных в файл на диске. Опция `-r 0` задаёт размер Bounding Box, равным размеру рисунка в пикселах. Так, если размер рисунка равен 600 px на 300 px, то размер Bounding Box будет 600 bp на 300 bp. При этом координаты точки отсчёта обычно не равны (0,0). В общем случае опция `-r` имеет вид `-r` *dpi*, где *dpi* — число точек на дюйм.

[5] Можно вставлять в документ чёрно-белые рисунки типа чертежей прямо в формате PCX, но качество изображения в этом случае, мягко говоря, неважное.

[6] 72 больших пункта (bp) равны 1 дюйму или 2.54 см

Так, если в случае рисунка размером 600 px на 300 px задать опцию в виде `-r 600`, то размер Bounding Box будет 72 bp на 36 bp или 1 дюйм (600/600) на 0.5 дюйма (300/600).

Программа `tiff2ps.ps` из набора `pstools` с помощью `GhostScript`'а преобразует файл в формате TIFF в `eps`-файл. Формат TIFF позволяет хранить рисунки в сжатом виде (обычно используется метод LZW). `eps`-файл, полученный из такого `tif`-файла, может иметь достаточно большой размер. Программа `tiff2ps.ps` позволяет избежать хранение одного и того же рисунка сразу в двух файлах. Если использовать опцию `-h`, то в заголовке `eps`-файла будет записана ссылка на `tif`-файл, а сам рисунок не будет копироваться туда. Вот типичный пример такой командной строки:

`gswin32c -- tiff2ps.ps` *inputfile*`.tif` *outputfile*`.eps -i -h -b0`

3.2 Драйвер `pdftex`

pdfLaTeX не поддерживает формат EPS. Зато он позволяет включать в документ графику в векторном формате PDF[7]. Преобразовать рисунок из EPS в PDF можно, например, с помощью программы `GSview`. Но обычно конверторы не задают в `pdf`-файле параметры CropBox'а (аналог BoundingBox формата EPS). Поэтому pdfTeX при включении рисунка в документ использует параметры MediaBox'а, задающие размер листа с рисунком, а не размер самого рисунка. Чтобы получить правильный Bounding Box из `eps`-файла, надо перед конвертирование преобразовать `eps`-файл так, чтобы точкой отсчёта стала точка (0,0) и чтобы размер страницы точно соответствовал BoundingBox. Такую работу делает программа `epstopdf.exe`. В командной строке надо задавать только имена входного `eps`-файла и выходного `pdf`-файла:

`epstopdf` *inputfile*`.eps` *outputfile*`.pdf`

Так, в файле `a.pdf`, полученном из файла `a.eps` с помощью `epstopdf`, MediaBox задан как `[0 0 56 56]`.

Драйвер `pdftex`, в отличие от `dvips`, поддерживает включение в документ рисунков в растровых форматах PNG, TIFF и, что особенно важно для фотографических изображений, в формате JPEG.

4 Вставка рисунка из графического файла

В пакете graphicx определена команда

`\includegraphics[`*keyval-list*`]{`*file*`}`

для вставки в документ рисунка из графического файла *file*. Необязательный аргумент *keyval-list* может содержать целый список ключей. Значения ключей задаются

[7] Поддерживается также включение рисунков из файлов, созданных MetaPost'ом. Как это делается, можно найти в документации к pdfTeX'у.

в виде *key=value*, а в списке они перечисляются через запятую.

Опцию команды `\includegraphics` можно опустить, если мы хотим вставить в документ рисунок в «натуральную» величину из `eps`-файла в случае драйвера `dvips` или из `pdf`-файла в случае драйвера `pfdtex`. TEX выделяет в документе под рисунок бокс, размер которого считывается из BoundingBox и MediaBox, соответственно. Пример[8]:

```
\includegraphics{a}
```

Видно, что точкой отсчёта бокса, выделенного под рисунок, является его левый нижний угол. В этом примере расширение файла с рисунком опущено. Это сделано для того, чтобы входной файл можно было обрабатывать как LATEX'ом, так и pfdLATEX'ом: LATEX берёт рисунок из файла `a.eps`, а pfdLATEX — из файла `a.pdf`. Более подробно читайте об этом в разделе 4.4.

BoundingBox в `eps`-файле задаёт размер рисунка с точностью до 1 bp. Бывают случаи, когда такой точности недостаточно. Тогда можно использовать параметры из строки (если она присутствует в файле)

`%%HiResBoundingBox:` *llx lly urx ury*

в которой числа с десятичной точкой *llx*, *lly*, *urx* и *ury* (они имеют тот же смысл, что и для BoundingBox) задают размер рисунка с более высокой точностью, чем 1 bp. Для того, чтобы TEX читал именно эту строку, надо в команде `\includegraphics` указать ключ

`hiresbb`

4.1 Как задать размер рисунка в документе

В этом разделе описаны ключи команды `\includegraphics`, с помощью которых можно задать размер прямоугольный области, выделяемой для размещения рисунка в документе.

Ключ

`width=`*length*

устанавливает значение *length* (в любых TEX'овских единицах длины) в качестве

[8] Кроме рисунка (он выполнен синим цветом), в этом и следующих примерах показаны также базовая линия строки с рисунком и границы бокса, отведённого под рисунок.

ширины области, выделяемой для размещения рисунка. Пример:

```
\includegraphics[width=1.5cm]{a}
```

Ключ

`height=`*length* | `totalheight=`*length*

устанавливает значение *length* (в любых TEX'овских единицах длины) в качестве ширины (полной ширины) области, выделяемой для размещения рисунка. Пример:

```
\includegraphics[width=1in,height=10mm]{a}
```

Ключ

`keepaspectratio`

обеспечивает сохранение отношения ширины к высоте самого рисунка, если заданные значения ширины и высоты области, выделенной под рисунок, нарушают, как в предыдущем примере, это отношение. Пример:

```
\includegraphics[width=1in,height=1cm,%
keepaspectratio]{a}
```

Видно, что в этом примере размер бокса, отведённого под рисунок, задаётся значением ключа `height`. Ключ `width` просто игнорируется: иначе рисунок вышел бы за пределы выделенной под него области по вертикали.

Ключ

`scale=`*scale*

изменяет «натуральный» размер рисунка в *scale* раз. Пример:

```
\includegraphics[scale=0.5]{a}
```

Результат, аналогичный действию описанных выше ключей, можно получить с помощью команд изменения размеров боксов, описанных в разделах 5.1.1 и 5.1.2. Пример:

```
\resizebox{1in}{10mm}{\includegraphics{a}}
```

Для полноты изложения материала надо упомянуть о мало пригодном для драйверов `dvips` и `pdftex` ключе

`bb=`*llx lly urx ury*

Этот ключ в виде `bb=0 0` *W H* приходится использовать при вставке в документ растрового рисунка в формате PCX (см. примечание на стр. 11). Здесь *W* и *H* — ширина и высота бокса, в который LaTeX помещает рисунок. pdfLaTeX при использовании ключа `bb` выдаёт предупреждение.

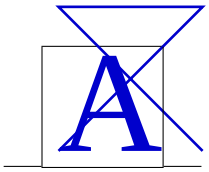
4.2 Включение в документ части рисунка

Ключи

`viewport=`*llx lly urx ury*
`trim=`*dl db dr du*

задают так называемую видимую область рисунка. Именно под эту часть рисунка отводится место в документе. Здесь *llx*, *lly*, *urx* и *ury* — это x- и y-координаты (в любых TeX'овских единицах длины) левого нижнего и правого верхнего углов видимой области рисунка относительно точки отсчёта, а *dl*, *db*, *dr* и *du* — это расстояния (в любых TeX'овских единицах длины) между левыми, нижними, правыми и верхними границами видимой области рисунка и самого рисунка, соответственно. Отрицательные значения смещения допустимы. Пример:

```
\includegraphics[viewport=-5 -5 40 40]{a}
```



Ключ

`clip=`*boolean*

отсекает часть рисунка, выходящую за границы видимой области, если значение *boolean* равно `true` (или просто указано). Пример:

```
\includegraphics[trim=-5 -5 16 16,clip]{a}
```

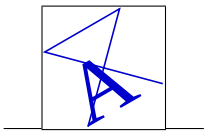
4.3 Поворот рисунка

Ключ

`angle=`*angle*

поворачивает рисунок на *angle* градусов против часовой стрелки. По умолчанию ось вращения проходит через точку отсчёта бокса. Пример:

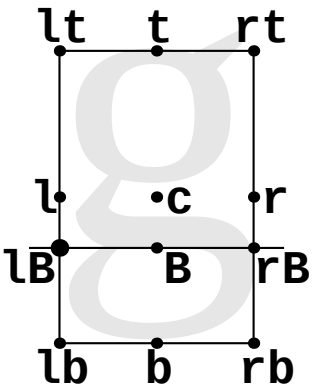
```
\includegraphics[scale=0.6,angle=30]{a}
```



Ключ

`origin=`*pos*

позволяет указать одно из 12 предопределённых положений оси вращения, показанных на следующем рисунке:

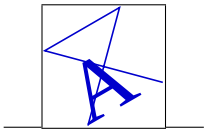


Положения точек вращения и допустимые значения *pos*.

Положение оси по горизонтали задаётся одной из трёх букв: `l` (на левой стороне бокса), `c` (по центру бокса) или `r` (на правой стороне бокса). Положение оси вращения по вертикали задаётся одной из четырёх букв: `b` (на нижней стороне бокса), `B` (на базовой линии), `c` (по центру бокса) или `t` (на верхней стороне бокса). В результате получается следующий список допустимых значений *pos*: `lb`, `lB`, `l`, `lt`, `b`, `B`, `c`, `t`, `rb`, `rB`, `r`, `rt`. Если указана только одна буква, то второй предполагается `c`. Так, `origin=lB` соответствует вращению вокруг точки отсчёта, в `origin=c` — вокруг центра бокса.

Результат, аналогичный действию описанных в этом разделе ключей, можно получить с помощью команды вращения боксов, описанной в разделе 5.3:

```
\rotatebox{30}{\scalebox{0.6}%
{\includegraphics{a}}}
```



На память Порядок следования ключей поворота и установки нового размера бокса через исходные размеры бокса влияет на результат, поскольку поворот меняет такие параметры бокса, как `\width` и др. (см. пример на странице 22). Ключи читаются слева направо!

4.4 Имена файлов без расширения

Расширение файла с рисунком в команде `\includegraphics` можно не указывать, поскольку драйвер сам знает, какие типы файлов он может обработать, а какие нет. Для драйвера `dvips` это файлы с расширением

`eps`, `ps`, `eps.gz`, `ps.gz`, `eps.Z`

а для драйвера `pdftex`

`png`, `pdf`, `jpg`, `mps`, `tif`

Когда расширение файла в команде `\includegraphics` не указано, драйвер последовательно (слева направо по списку) добавляет к имени файла все известные ему расширения и ищет файл уже по полному имени. Поиск прекращается, как только находится первый подходящий файл.

Пакет graphicx вводит декларацию

`\DeclareGraphicsExtensions{`*ext-list*`}`

которая позволяет вместо списка по умолчанию задать свой собственный список расширений, которые драйвер будет добавлять к имени файла во время его поиска. В *ext-list* расширения перечисляются через запятую, причем перед названием расширения ставится точка. Расширения используются в порядке их перечисления в списке.

На заметку Опуская в команде `\includegraphics` расширение файла с рисунком, можно один и тот же входной файл обрабатывать как LaTeX'ом, так и pfdLaTeX'ом.

4.4.1 Нестандартные расширения

На заметку Драйвер `dvips` трактует все файлы с неизвестными ему расширениями как рисунки типа `eps`. Поэтому вы можете использовать для своих `eps`-файлов *любые* нестандартные расширения.

4.5 Местоположение файлов с рисунками

По умолчанию LaTeX ищет файлы с рисунками в каталоге, в котором находится входной файл. Декларация

`\graphicspath{`*dir-list*`}`

позволяет расширить область поиска. В списке *dir-list* каждая из директорий заключается в фигурные скобки. Если указать, например

```
\graphicspath{{images/}{d:/images/eps/}{d:/images/pdf/}}
```

то LaTeX будет искать файлы с рисунками также в подкаталоге `images` текущего каталога (где находится входной файл) и в директориях `d:/images/eps` и `d:/images/pdf`.

На заметку Поскольку `eps`-файл является текстовым файлом, мы можем хранить его прямо во входном файле в окружении `filecontents*` из стандартного LaTeX'а.

4.6 «Нестандартные» файлы с рисунком

На память Всё, описанное в этом разделе, относится только к драйверу `dvips` и *не поддерживается* драйвером `pfdtex`.

Драйвер `dvips` умеет включать в документ рисунки из `eps`-файлов, которые сжаты программой `gzip`. Они распознаются по расширению `eps.gz`. Вот как выглядит предписание о трактовке файлов с таким расширением в файле `dvips.def`:

```
\@namedef{Gin@rule@.eps.gz}#1{{eps}{.eps.bb}{`gunzip -c #1}}
```

Здесь `#1` заменяет собой имя файла *name*. Итак, в предписании сказано, что файл с расширением `eps.gz` содержит рисунок типа `eps`, размер рисунка надо читать в файле *name*`.eps.bb`, а перед включением рисунка в документ надо сначала распаковать файл *name*`.eps.gz` программой `gunzip`. Файл *name*`.eps.bb` с размером рисунка нужен TeX'у на стадии вёрстки документа. Он должен состоять из строки

`%%BoundingBox:` *llx lly urx ury*

взятой из `eps`-файла (её смысл описан на странице 11). Дело в том, что сам файл с рисунком сжат и TeX не может прочитать из него размер рисунка.

С помощью декларации

`\DeclareGraphicsRule{`*ext*`}{`*type*`}{`*read-file*`}{`*command*`}`

можно: (i) связать расширение *ext* с типом рисунка *type*, понятным драйверу (`eps` для `dvips`); (ii) указать расширение файла *read-file*, из которого драйвер может получить данные о размере рисунка; (iii) задать команду *command*, которая должна быть выполнена перед включением рисунка в документ. Аргументы декларации `\DeclareGraphicsRule` имеют тот же смысл, что и в приведенной выше инструкции для расширения `eps.gz` из файла `dvips.def`. Перепишем эту инструкцию, используя декларацию `\DeclareGraphicsRule`:

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

Подчеркнём, что в аргументах *ext* и *read-file* перед названием расширения ставится точка.

Приведём теперь пример с декларацией `\DeclareGraphicsRule` по включению в документ фотографического изображения солнечного затмения из файла `s.jpg`. На странице 11 рассказано, как преобразовать рисунок из «чужого» для `dvips` формата JPEG в формат EPS. Итак, преобразуем сначала файл `s.jpg` в файл `s.eps`, затем создадим текстовый файл `s.jpg.bb` и скопируем в него из `eps`-файла строку с BoundingBox. Теперь файл `s.eps` можно удалить. Напечатаем во входном файле строку

```
\DeclareGraphicsRule{.jpg}{eps}{.jpg.bb}{`jpeg2ps -r 0 -h #1}
```

Теперь драйвер `dvips` знает, что ему надо делать с `jpg`-файлами: вызвать сначала программу `jpeg2ps`, которая преобразует рисунок из `jpg`-файла в формат EPS (без записи на диск, поскольку опция `-o` опущена), а затем вставить рисунок в документ. TEX'у на стадии вёрстки документа требуются параметры BoundingBox. Он возьмёт их из файла `s.jpg.bb`. После этого, команда

```
\includegraphics[width=3cm]{s.jpg}
```

во входном файле не вызовет ошибки ни LATEX'а, ни pdfLATEX'а:

```
\includegraphics[width=3cm]{s.jpg}
```

Вместо декларации `\DeclareGraphicsRule` можно воспользоваться ключами

`ext=`*ext*
`type=`*type*
`read=`*read-file*
`command=`*command*

команды `\includegraphics`. Ключи задают те же правила, что и аргументы декларации `\DeclareGraphicsRule`. Описанный выше пример с включением в документ рисунка из файла `s.jpg` при использовании ключей перепишется как

```
\includegraphics[width=3cm,%
  type=eps,read=.bb,command=`jpeg2ps -r 0 -h #1]{s.jpg}
```

или как

```
\includegraphics[width=3cm,%
  type=eps,ext=.jpg,read=.jpg.bb,command=`jpeg2ps -r 0 -h #1]{s}
```

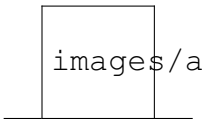
4.7 Черновой режим

На стадии подготовки документа можно использовать ключ

`draft`

который указывает, что вместо рисунка надо начертить рамку и напечатать внутри неё имя файла с рисунком. Пример:

```
\includegraphics[width=1.5cm,draft]{a}
```



Черновой режим обычно задаётся для всех рисунков сразу опцией `draft` либо в команде загрузки пакета, либо в `\documentclass`. В этом случае ключом

`final`

в команде `\includegraphics` можно отметить черновой режим для некоторых рисунков.

5 Манипуляции с блоками

Ключи команды `\includegraphics` позволяют манипулировать боксом с рисунком. Пакет graphicx позволяет нам любую часть страницы документа оформить в виде бокса и манипулировать им, поворачивая, растягивая или сжимая его произвольным образом.

5.1 Изменение размеров бокса

5.1.1 Трансформация к указанному размеру

Команда

`\resizebox{`*width*`}{`*height*`}{`*lr-text*`}`

помещает, подобно команде `\mbox`, текст *lr-text* в бокс и затем сжимает или растягивает бокс (вместе с его содержимым) так, чтобы его ширина и высота стали равны *width* и *height*:

```
\resizebox{4cm}{8mm}{abc}
```

abc

Сохранить отношение высоты к ширине бокса можно, указав в качестве *width* или *height* восклицательный знак `!`:

```
\resizebox{3cm}{!}{abc}
```

Команда

`\resizebox*{`*width*`}{`*totalheight*`}{`*lr-text*`}`

действует подобно команде `\resizebox`, но подгоняет не высоту, а полную высоту бокса (сумма высоты и глубины) к указанному размеру *totalheight*.

На заметку В аргументах команды `\resizebox` исходные размеры бокса доступны в виде командных длин `\height`, `\width`, `\totalheight` и `\depth`.

5.1.2 Трансформация по указанному масштабу

Команда

`\scalebox{`*h-scale*`}[`*v-scale*`]{`*lr-text*`}`

помещает, подобно команде `\mbox`, текст *lr-text* в бокс и затем изменяет его ширину в *h-scale* раз и высоту в *v-scale* раз. Если опция *v-scale* опущена, то отношение ширины к высоте бокса при трансформации блока сохраняется. При отрицательных значениях аргументов происходит зеркальное отражение текста:

```
\scalebox{1.5}[3]{abc} \scalebox{1.5}%
[-3]{abc}
```

5.2 Зеркальное отражение блока

Команда

`\reflectbox{`*lr-text*`}`

эквивалентна `\scalebox{-1}[1]{`*lr-text*`}`, описанной в разделе 5.1.2.

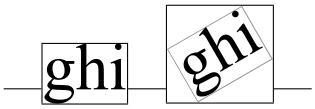
5.3 Поворот блока

Команда

`\rotatebox[`*keyval-list*`]{`*angle*`}{`*lr-text*`}`

помещает, подобно команде `\mbox`, текст *lr-text* в бокс и поворачивают его на *angle* градусов против часовой стрелки. По умолчанию бокс поворачивается относительно своей точки отсчёта[9]:

```
{\huge ghi \rotatebox{30}{ghi}}
```



Ключи *keyval-list* в необязательном аргументе команды `\rotatebox` позволяют изменить положение оси, вокруг которой поворачивается бокс, и единицу измерения угла поворота. Ключи в списке перечисляются через запятую и имеют вид *key=value*.

Ключ

`origin=`*pos*

позволяет указать одно из 12 предопределённых положений оси вращения, описанных в разделе 4.3.

Произвольное положение оси вращение можно задать с помощью ключей

`x=`*dimen1*
`y=`*dimen2*

Например, если `[`*keyval-list*`]` имеет вид `[x=4mm,y=3mm]`, то ось вращения проходит через точку, которая смещена относительно точки отсчёта бокса на 4 мм вдоль оси абсцисс и на 3 мм вдоль оси ординат.

В ключе

`units=`*number*

значение *number* соответствует полному обороту вокруг оси. Так, по умолчанию угол поворота измеряется в градусах, что соответствует `units=360`. Если задать `units=1`, то для поворота на 90 градусов надо в первом обязательном аргументе команды указать `0.25`. Пример:

```
\rotatebox[origin=c,units=1]{0.25}{\huge W}
```

[9] В этом и в следующем примере показаны базовая линия строки и границы боксов, занимаемых текстом до и после поворота.

6 Альбомная ориентация страницы

Пакет lscape вводит окружение

`\begin{landscape} ... \end{landscape}`

содержание которого печатается на странице в альбомной ориентации. Колонтитулы и подстрочные примечания печатаются как на обычных страницах. При смене портретной ориентации страницы на альбомную и назад остаются не до конца заполненные страницы. Не работает внутри плавающих объектов.

7 Общие настройки графического пакета

В этом разделе приведены опции пакета graphicx и рассказано, как можно задавать значения ключей сразу для нескольких команд.

7.1 Опции пакета

`hiderotate` указывает, что не надо отображать в документе повёрнутые боксы.

`hidescale` указывает, что не надо отображать в документе боксы с изменённым размером.

`hiresbb` указывает, что надо читать параметры `%%HiResBoundingBox` вместо `%%BoundingBox` (см. описание ключа `hiresbb` на стр. 13).

`draft | final` включает (выключает) черновой режим вёрстки документа. В черновом режиме вместо рисунков рисуются рамки.

7.2 Установка ключей

Пакет graphicx загружает пакет keyval, который вводит декларацию

`\setkeys{`*operation*`}{`*keyval-list*`}`

позволяющую установить одни и те же значения ключей в опции *keyval-list* сразу нескольких команд `\rotatebox` (при значении *operation* равным `Grot`) и `\includegraphics` (при значении *operation* равным `Gin`). Так,

```
\setkeys{Gin}{width=\textwidth}
```

устанавливает ширину всех рисунков в области действия декларации `\setkeys`, равной ширине текста на странице.

8 Навигационные элементы в документе PDF

Пакет hyperref создаёт в pdf-документе гипертекстовую навигацию и закладки (bookmarks), которые можно использовать при просмотре документа в Acrobat Reader. Этот пакет рекомендуется загружать последним, поскольку он переопределяет многие команды LaTeX'а. Для документов на русском языке надо указывать опцию unicode при загрузки пакета:

```
\usepackage[colorlinks,unicode]{hyperref}
```

Иначе в закладках кириллические символы будут пропущены. Если не указать опцию colorlinks, то гипертекстовые ссылки будут чёрного цвета и в рамке. Полный список опций можно найти в файле manual.pdf (автор Sebastian Rahtz), который входит в документацию пакета.

Пакет hyperref автоматически превращает *все* перекрёстные ссылки LaTeX'а в гипертекстовые ссылки. Гипертекстовыми ссылками становятся все элементы оглавления и списков таблиц и рисунков. Кроме того, все номера страниц в предметном указателе также становятся гипертекстовыми ссылками (опция hyperindex задана по умолчанию; её можно отключить, присвоив значение false).

Аргументы всех команд секционирования, которые используются при составлении оглавления, автоматически оформляются как закладки (опция bookmarks задана по умолчанию). Закладками становятся также записи, которые добавлены в оглавление командой \addcontentsline. Её надо использовать очень аккуратно. Дело в том, что LaTeX ставит в оглавлении номер страницы, на которой находится команда \addcontentsline. Но поскольку она не создаёт мишень для ссылки, то переход происходит не на это место, а на начало раздела, в котором находится команда. Именно команды секционирования являются мишенями. Такая проблема не возникает, когда команда \addcontentsline используется для записи в оглавление имён команд секционирования со звездочкой и располагается сразу *после* них.

С помощью команд \pdfbookmark и \hypertarget мы может создавать закладки сами. Как это делается, можно прочитать в файле paper.pdf (автор статьи Heiko Oberdiek), который входит в документацию пакета.

Пакет hyperref позволяет создавать гипертекстовые ссылки на любые источники, имеющие URL. Вы можете прочитать об этом в файле manual.pdf (автор Sebastian Rahtz), который входит в документацию пакета. Здесь я отмечу только команду

\href{*URL*}{*link text*}

Смысл аргументов ясен из следующего примера:

```
\href{mailto:syutkin@ns.kinetics.nsc.ru}%
{Адрес для контактов}
```

Адрес для контактов

Предметный указатель