LaTeX, GNU/Linux и русский стиль.

© Е.М. Балдин*

Эта статья была опубликована в июльском номере русскоязычного журнала Linux Format (`http://www.linuxformat.ru`) за 2007 год. Статья размещена с разрешения редакции журнала на сайте `http://www.inp.nsk.su/~baldin/` и до января месяца все вопросы с размещением статьи в других местах следует решать с редакцией Linux Format. Затем все права на текст возвращаются ко мне.

Текст, представленный здесь, не является точной копией статьи в журнале. Текущий текст в отличии от журнального варианта корректор не просматривал. Все вопросы по содержанию, а так же замечания и предложения следует задавать мне по электронной почте `mailto:E.M.Baldin@inp.nsk.su`.

Текст на текущий момент является просто *текстом*, а не книгой. Поэтому результирующая доводка в целях улучшения восприятия текста не проводилась.

*e-mail: E.M.Baldin@inp.nsk.su

Эмблемы TeX и METAFONT, созданные Дуайном Бибби, взяты со странички Д.Э. Кнута. Цветной пингвин взят из пакета `ps2pdf` от Ральфа Найпрашека (Rolf Niepraschk)

Оглавление

Глава 11

Начала программирования

> Когда придумываешь что-то сам, высок шанс ничего не придумать. Но когда живёшь чужим умом, уж точно ничего не придумаешь. Никогда не делай того, что делают другие. Это на 100% обрекает на неудачу.
>
> Герш Ицкович Будкер.

LaTeX позволяет не просто набирать текст — он позволяет его программировать, а, следовательно, перекладывать часть своей работы на компьютер. Привычка думать — одна из самых необычных особенностей разумного человека. Она позволяет экономить силы и время.

11.1. Создаём свои ...

... команды, окружения и прочее. Наверняка, возникшая в процессе набора, простенькая надоедливая проблема решена и не один раз. С другой стороны при нарастающей квалификации проще бывает изобрести этот велосипед заново в удобной на текущий момент форме:

```
\newcommand{\ee}{\ensuremath{e^{+}e^{-}}\xspace}
```

Часто новые команды создаются для комбинаций используемых исключительно в математическом окружении. Команда **\ensuremath** обеспечивает это окружение не зависимо от текущего режима:

```
\(J/\psi\to\ee\) является одним из
подвидов \ee-рассеяния.
```

$J/\psi \to e^+e^-$ является одним из подвидов e^+e^--рассеяния.

Команда \xspace из одноимённого пакета добавляет в конце команды пробел в случае, если за командой нет знаков препинания, то есть избавляет от необходимости самому вставлять явный пробел после команды.

Имеются три команды, которые позволяют создавать свои или переименовать уже имеющиеся макросы:

---

newcommand{«команда»}[N][«зн. по ум.»]{«определение»}
renewcommand{«команда»}[N][«зн. по ум.»]{«определение»}
providecommand{«команда»}[N][«зн. по ум.»]{«определение»}

---

newcommand определяет новую команду. Если такая команда уже была, то при компиляции генерится ошибка. **renewcommand** напротив переопределяет уже существующую команду. В свою очередь \providecommand создаёт новую команду, если на момент описания такой команды не было, и ничего не делает, если она уже существовала.

В каждом из этих макросов есть два обязательных параметра — это имя команды и её описание. Если команде необходимо передать параметр/параметры, то первый необязательный аргумент должен принять значение от одного (1) до девяти (9). В разделе 3.4 обсуждалась команда для дублирования знака в формуле при переносе её на следующую строку (\(a + b \hm{=} c\)):

```
\newcommand*{\hm}[1]{#1\nobreak\discretionary{}%
  {\hbox{$\mathsurround=0pt  #1$}}{}}
```

Вместо знака решётки (#) с цифрой после него при компиляции макроса подставляется соответствующий параметр. В данном случае параметр был только один и можно сказать, что его значение сохраняется в «переменной» #1.

Звёздочка (*) в конце макроса **newcommand** налагает на передаваемый параметр команды \hm дополнительное условие: в передаваемом тексте не должно быть пустых строк и команды **par**. В некоторых случаях это упрощает отладку кода.

Наличие второго необязательного параметра в макросах определения новых команд позволяет первый параметр создаваемой команды определить как параметр по умолчанию:

```
\newcommand{\exmpl}[1][умолчанию]%
                  {<<значение по #1>>}
Сравните \exmpl{} и \exmpl[требованию].
```

Сравните «значение по умолчанию» и «значение по требованию».

Для определения нового окружения используется команда **newenvironment**, например:

```
\newenvironment{outlined}{\hrule\begin{center}}%
                 {\end{center}\smallskip\hrule}
\begin{outlined}
  Выделенный текст.
\end{outlined}
```

Выделенный текст.

Формальное описание этой команды похоже на описание **newcommand**:

newenvironment{«окружение»}[N][«зн. по ум.»]%
{«код открывающий окружение»}%
{«код закрывающий окружение»}

Точно так же, как и в случае **newcommand**, созданному окружению можно передавать параметры. Подставлять параметры можно только в коде открывающему окружение. Кроме команды создания нового окружения можно так же переопределять уже имеющиеся с помощью аналогичной команды **renewenvironment**.

В разделе 8.3, посвящённом описанию презентационного класса **beamer** упоминалось об ещё одной возможности создавать новые именованные окружения с помощью команды **newtheorem**:

```
\newtheorem{Texmpl}{Пример}
\begin{Texmpl}[Теорема Пифагора]\label{th:1}
  Пифагоровы штаны во все стороны равны.
\end{Texmpl}

\begin{Texmpl}\label{th:2}
  Мудрость ограничена, а глупость бесконечна.
\end{Texmpl}

Можем сослаться первую теорему:~\ref{th:1},
а можно и на вторую:~\ref{th:2}
```

Пример 1 (Теорема Пифагора). *Пифагоровы штаны во все стороны равны.*

Пример 2. *Мудрость ограничена, а глупость бесконечна.*

Можем сослаться первую теорему: 1, а можно и на вторую: 2

Команда **newtheorem** имеет две формы:

newtheorem{«теорема»}[«существующая теорема»]{«заголовок»}
newtheorem{«теорема»}{«заголовок»}[«имя счётчика»]

Каждая из форм имеет по два соответствующих обязательных аргумента и одному необязательному. В первом случае это имя уже существующей теоремы с которой следует иметь совместную нумерацию. Во втором случае в качестве необязательного параметра передаётся имя уже существующего счётчика на основе которого строится нумерация. О том, что такое счётчики и как их определять, речь пойдёт далее.

11.2. Счётчики и другие переменные

«Другие переменные» уже обсуждались в разделе 6.1 „Определённые «размеры» и переменные «длины»”. Операции с этими переменными выполнялись с помощью команд **newlength**, \setlength и **addtolength**. Аналогично в LaTeX представлена и целочисленная арифметика с использованием счётчиков в качестве переменных:

```
\newcounter{MyCount}\setcounter{MyCount}{5}
Значение MyCount равно \arabic{MyCount},
или~\alph{MyCount}, или  \Asbuk{MyCount}.\par
\addtocounter{MyCount}{1550}
\arabic{MyCount} эквивалентно \Roman{MyCount}.
```

Значение MyCount равно 5, или e, или Д.

1555 эквивалентно MDLV.

Новый счётчик создаётся с помощью команды **newcounter**. При создании новый счётчик инициализируется нулём. Создание счётчика является глобальной операцией, то есть при компиляции информация о его создании не исчезнет, даже если новый счётчик был определён внутри окружения. Для присвоения счётчику другого значения используется команда \setnewcounter, а для изменения на какое-то определённое число — **addtocounter**.

В отличии от длин, основная роль которых помнить размеры какого-то определённого бокса, счётчики используются для отображения какой-либо структурной информации. Поэтому особое внимание уделяется написанию счётчиков. Чтобы просто отобразить численное значение счётчика с помощью арабских цифр используется команда **arabic**{счётчик}. Для римской числовой нотации необходимо воспользоваться командой **Roman** и **roman** — заглавные и строчные буквы, соответственно. Счётчик может быть представлен так же буквой алфавита: **alph** — латинская строчная, \asbuk — кириллическая строчная и \asbuk — кириллическая заглавная.

В стандартных классах уже определён набор счётчиков в которых хранятся номера страницы (счётчик **page**), раздела (соответственно, счётчики **part**, **chapter**, **section**, **subsection**, **subsubsection** и т. д.), подстрочного примечания (счётчик **footnote**), плавающих окружений (счётчики **figure** и **table**) и формул (**equation**). При создании счётчика также автоматически создаётся команда с префиксом **the** перед именем счётчика. Вызов такой команды выводит номер счётчика. При выводе номера раздела, плавающего объекта, уравнения и тому подобного используются именно такого рода команды, поэтому, переопределив **the**-команду, можно немного изменить стиль, например, следующая команда предписывает в дальнейшем маркировать все страницы в римском стиле:

```
\renewcommand{\thepage}{\Roman{page}}
```

На базе счётчиков можно организовывать иерархические структуры, то есть можно указывать зависимости:

```
\newcounter{Main}\addtocounter{Main}{10}
\newcounter{Dep}[Main]\addtocounter{Dep}{10}
Было: \theMain.\theDep\par
\stepcounter{Main}
Стало: \theMain.\theDep
```

Было: 10.10

Стало: 11.0

При создании нового счётчика можно создать связь с уже существующим, указав имя существующего счётчика в качестве необязательного параметра. В примере

выше счётчик **Dep** зависит от счётчика **Main**. Эта связь проявляется в том, что если увеличить значение базового счётчика (**Main**) на единицу с помощью команды **stepcounter**, то подчинённый счётчик (**Dep**) обнуляется. Обычно, новый счётчик устанавливают в подчинение счётчикам разделов (**section**).

Команда **refstepcounter**{счётчик} отличается от **stepcounter**, тем, что помимо обнуления всех зависимых счётчиков, **refstepcounter** определяет значение, выводимое командой ссылки **ref**, как текст, создаваемый **the**-командой:

```
% окружение "Задача"
\newcounter{Problem}[section]
\renewcommand{\theProblem}{\thesection.\arabic{Problem}}
\newenvironment{Problem}[0]{%
  \par\refstepcounter{Problem}%
  \theProblem\,}%
{\par}%
```

Здесь определено окружение **Problem** и одноимённый счётчик. Счётчик **Problem** зависит от счётчика раздела. Вывод счётчика \theProblem переопределён как номер раздела за которым следует уже сам счётчик. Внутри окружения счётчик **Problem** увеличивается на единицу с помощью команды **refstepcounter**{счётчик}. Результат использования нового окружения представлен в следующем примере:

```
\begin{Problem}\label{ex:1}
  Задача раз
\end{Problem}
\begin{Problem}\label{ex:2}
  Задача два
 \end{Problem}
Ссылки на раз~\ref{ex:1} и два~\ref{ex:2}.
```

11.2.1 Задача раз

11.2.2 Задача два

Ссылки на раз 12.2.1 и два 12.2.2.

При работе с переменными LaTeX также могут помочь следующие пакеты:

calc — макропакет из коллекции **tools** для арифметических вычислений, уже упоминавшийся в разделе 6.1. Этот пакет переопределяет команды типа **newcounter** так, что в них можно использовать арифметические выражения, хоть и с некоторыми ограничениями. Подробности в файле `calc.pdf`.

ifthen — макропакет в котором определена команды условного перехода \ifthenelse и цикла \whiledo. Подробности в файле `ifthen.pdf`. Так же можно присмотреться к усовершенствованной версии этого пакета **xifthen**.

fmtcount — представляет различные форматы (двоичный, восьмеричный, шестнадцатеричный и т. д.) отображения счётчиков (`fmtcount.pdf`).

multido — определяет оператор цикла \multido (`multido.pdf`).

tokenizer — позволяет разбивать тестовые списки на элементы (`tokenizer.pdf`).

totpages — даёт возможность узнать число страниц в документе и тому подобную информацию (`totpages.pdf`).

xkeyval — улучшенная версия пакета **keyval**, который позволяет передавать/принимать в качестве параметров пары значений «key=value» (`xkeyval.pdf`).

11.3. Создаём свой пакет

Предположим, что Вы уже владеете искусством программирования в среде LaTeX. Для того чтобы распространить свои наработки следует организовать исходники в удобном для дальнейшей поддержки, передаче и установке виде. Хотя можно и не стараться, если Вас не интересует результат.

Внимание: Политика создания названий команд в TeXподобной среде такова, что для новых пакетов необходимо придумывать новые команды. Это сделано для обеспечения абсолютной совместимости сверху вниз. К сожалению подобная политика в случае бездумного использования слов может привести «захватыванию» подходящих сочетаний[1].

Знать как правильно устроен пакет полезно и новичку, так как эффективное обучение программированию напрямую связано с изучением уже существующего кода.

В LaTeX сообществе принято распространять свои пакеты и документацию к ним в виде автономных файлов с расширением `dtx` (dtx-файлы). Для автоматической установки пакетов используются инструкции, записанные в файлах с расширением `ins` (ins-файлы). Для более подробной информации следует обратиться к инструкции «How to Package Your LaTeX Package», созданной Скотом Пакиным (Scott Pakin). Файл `dtxtut.pdf`, как обычно, можно найти в стандартной поставке LaTeX или на CTAN. Вместе с документацией идут файлы примеров `[c]skeleton.dtx` и `[c]skeleton.ins`.

За работу с dtx-файлами отвечает пакет **doc** и сопутствующая ему утилита DOCSTRIP (файл `docstrip.pdf`). Основная идея пакета **doc** состоит в совмещении кода с документацией, что облегчает поддержку и развитие пакета.

11.3.1. Установочный ins-файл

Для извлечения кода и документации из dtx-пакета следует написать специальный установочный файл. Набор инструкций достаточно стереотипен:

```
%% Стандартный копирайт по выбору (рекомендуется LPPL/GPL)
%%
%% Первый шаг - загрузка DOCSTRIP.
\input docstrip.tex
%% Подробный отчёт о каждом шаге хорош только когда пакет
%%отлаживается.
\keepsilent
```

[1] Примером может служить пакет **listings**, где вместо подходящего по названию окружения **listing** используется **lstlisting**.

```
%% Директория в которую устанавливается пакет. Имя
%%директории является относительным по отношению к
%%базовой директории $(TEXMF).
\usedir{tex/latex/{«имя пакета»}}
%% Определение преамбулы, которая вставляется во все
%%сгенерированные файлы. Обычно, этоинформация об авторе
%%и пожелания пользователям.
\preamble
 Текст преамбулы
\endpreamble
%% Извлечение файлов пакета из dtx. Основной шаг,
%%который может повторяться несколько раз.
\generate{\file{«извлекаемый файл»}{\from{«dtx-файл»}{метка}}}
   ...
   ...
%% Информация для пользователя. Всегда что–то полезно сказать
%%после установки.
\obeyspaces
\Msg{*****************************************************}
\Msg{*       Здорово, что Вы поставили этот пакет.       *}
\Msg{*    Прочитайте документацию перед использованием!  *}
\Msg{*****************************************************}
%% Метка конца установочного файла.
\endbatchfile
```

11.3.2. Пакетный dtx-файл

Пакетный dtx-файл содержит в себе и код с комментариями, и текст описания пакета. Структура dtx-файла после прогона через **latex** позволяет получить печатную документацию. Код с комментариями тоже может стать частью документации. Это шаг по направлению к «грамотному программированию» (literate programming).

Наличие комментариев в коде заставляет для получения результата повторять процедуру компиляции дважды. Первый раз отрабатывается LaTeX-код, а затем комментарии. Во втором случае знак % перед комментарием игнорируется и текст комментария передаётся на вход LaTeX, если он (комментарий) не окружён командами **iffalse**-**fi**.

Пролог

В начале следует, естественно, добавить информацию об авторе:

```
%\iffalse meta–comment
% Этот текст не обрабатывается LaTeX'ом. Слово meta–comment
%добавлено просто для удобства чтения кода человеком и
```

```
%означает, что этот текст предназначен именно для него (человека).
%\fi
```

В ins-файле в команде \generate использовался параметр «метка». Это говорит DocStrip, что следует выбирать строки, которые следуют за комментарием и конструкцией <метка> или между тегами <*метка> и </метка>. Далее идёт код заголовка пакета соответствующего метке «метка»:

```
% \iffalse
%<метка> \NeedsTeXFormat{LaTeX2e}
%<метка> \ProvidesPackage{«имя пакета»}
%<метка>         [<ГГГГ>/<ММ>/<ДД> v<версия> <краткое описание>]
```

Строчку «<ГГГГ>/<ММ>/<ДД> v<версия> <краткое описание>» нужно заменить на дату, версию и краткое описание, соответственно.

Закончить пролог необходимо следующими словами, создающими основную документацию:

```
%<*driver>
\documentclass{ltxdoc}
\usepackage{«имя пакета»}
\begin{document}
  \DocInput{«dtx-файл»}
\end{document}
%</driver>
% \fi
```

Это единственная часть относящаяся к документации, которая не начинается со знака комментария (%).

В прологе можно указывать ещё некоторое количество инструкций уточняющих формат создаваемой документации.

Пользовательская документация

Прежде всего следует учесть, что подавляющий объём описаний для пакетов LaTeX сделан на английском языке. Для этого есть довольно веские основания, связанные с размером англоязычной аудиторией.

Написание документации для dtx-пакета ничем не отличается от написания обычного LaTeX-документа за исключением, что не следует забывать о знаке комментария (%) в начале строки

```
% \title{Пакет \textsf{«имя пакета»}}
% \author{«Ваше имя» \\ \texttt{«Ваш e-mail»}}
% \maketitle
% текст документации
```

К стандартным LaTeX-командам секционирования уровня **paragraph** добавляются \DescribeMacro{макрос} и \DescribeEnv{окружение}

Код с комментариями

Очевидно, что лучшая документация для программиста это сам код, но для нормального человека описательный текст предпочтительнее. Проблема совмещения кода и описания является основной причиной возникновения «грамотного программирования» (literate programming).

После окончания пользовательской документации идёт сам код:

```
%\StopEventually{\PrintIndex}
%\begin{environment}{«имя окружения»}
% Описание окружения.
% Аналогично, существует окружение macros, для описания новых команд.
%␣␣␣␣\begin{macrocode}
 Здесь идёт код, вида:
\newenvironment{«имя окружения»}{начало}{окончание}
%␣␣␣␣\end{macrocode}
%\end{environment}

 ...

%\Finale
\endinput
```

Команда \StopEventually{} отмечает начало кода и принимает в качестве параметра команду, которую следует выполнить в конце документации, например, распечатать алфавитный указатель \PrintIndex.

Любой код следует окружать с помощью окружения **macrocode**. Это позволит включить его в печатную документацию. Есть две особенности, для этого окружения, которые следует учитывать:

- между % и **begin**{macrocode} должно быть ровно четыре (4) пробела. Аналогичное правило действует и для **end**{macrocode},
- внутри этого окружения не должно быть текста, начинающегося с %.

Внутри окружений **environment** и **macros** может быть несколько вставок кода и текста.

11.3.3. Пакетирование

Часто LaTeX-пакеты распространяются в виде одного dtx-файла. Существует способ включить установочный ins-файл в файл пакета:

```
%<*batchfile>
\begingroup

   Содержание ins-файла
```

```
\endgroup
%</batchfile>
```

Следует только убрать заключительную команду \endbatchfile, для того чтобы LaTeX мог скомпилировать всё остальное.

Всё. Теперь для распространения свой пакет лучше всего поместить на CTAN. Для закачки следует обратиться к ресурсу `http://www.ctan.org/upload`. Всегда необходим краткий `README` с описанием. Собранная документация в виде pdf-файла так же является хорошим тоном.

11.4. Напутствие

Документируйте каждый шаг. Пишите как можно больше качественного текста, так как его мало не бывает. Живучесть программы определяется не только кодом, но и описанием. «Светлое будущее» за грамотным программированием (literate programming).

Вот и закончился LaTeX-цикл в журнале. Честно говоря, я сам за это время узнал много чего нового для себя. Надеюсь мне удалось поделиться этими знаниями с читателями. В этой информации нет никакой чёрной магии — всё просто и логично. Эта информация полезна, так как позволяет автоматизировать одно из самых сложных ремёсел человеческой цивилизации — создание книг. Пишите тексты большие и маленькие — они не пропадут.