

Продвинутый L^AT_EX

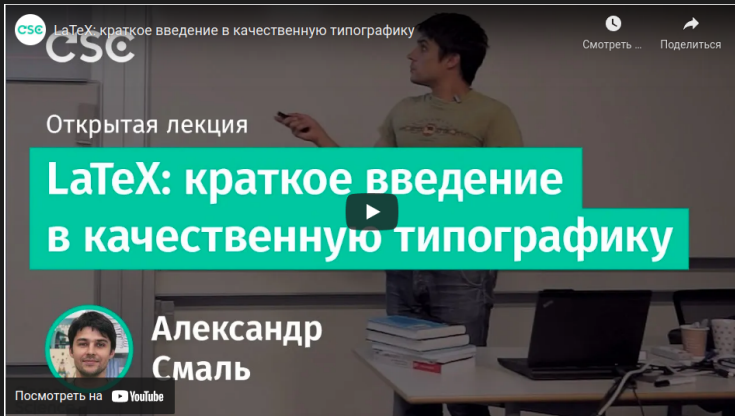
Типография и работа с командами

Антон Лиознов

CSC

2022

Предыстория



|| Предыстория

- ▶ В 2015 году в CSC была лекция “LaTeX: краткое введение в качественную типографику” от Александра Владимировича Смаля
- ▶ Она знакомила с основами \LaTeX и показывала прелесть этой системы
- ▶ Если вы видите эти строки в записи и лишь начинаете изучать \LaTeX , советую ознакомиться с лекцией
- ▶ И прочитать книгу Львовского “набор и верстка в системе \LaTeX ”

Что сегодня узнаем?

1. Прimitives в \LaTeX

- 1.1 Как \TeX видит наш документ: боксы и клей
- 1.2 Какие primitives существуют: длины, счётчики и другое
- 1.3 Как манипулировать primitives

2. Программирование в \TeX и \LaTeX

- 2.1 Создание макросов
- 2.2 Условные операторы
- 2.3 Циклы и рекурсия
- 2.4 Операции ввода-вывода
- 2.5 Отладка

Эта лекция для тех, кто уже знает \LaTeX хотя бы на уровне книги Львовского

Для чего вам эти знания



- ▶ Скорее всего не понадобится
- ▶ Документы можно составлять и из готовых шаблонов

Для чего вам эти знания



- ▶ Скорее всего не понадобится
- ▶ Документы можно составлять и из готовых шаблонов



- ▶ Для общей эрудиции
- ▶ “подправлять” используемые шаблоны
- ▶ писать свои шаблоны
- ▶ автоматизировать работу

Обо мне

- ▶ закончил CSC в 2015 году
- ▶ стажировался в Paragoria, онлайн L^AT_EX-редакторе
- ▶ младший научный сотрудник в Сколковском институте науки и технологий

Некоторые соглашения

Сноски

- ▶ Для повторного прочтения
- ▶ Некоторые детали по работе с командами
- ▶ Ссылки на источники
- ▶ Комментарии
- ▶ будут видны вне класса



Некоторые соглашения

“магические” слайды

Слайды с дополнительной информацией

- ▶ Для полноты картины
- ▶ Но не для анализа в классе

|| Некоторые соглашения

Слайды только для класса

Такие слайды исчезнут в выкладываемой лекции.

- Так обозначаем сноску, которая будет только в классе
- ← видна

|| Некоторые соглашения

Слайды только для чтения

Такие слайды появятся в выкладываемой лекции.

- Так обозначаем сноску, которая будет только вне
- ← классе видна

Задача – сделать шаблон выступлений

Наша практическая задача сегодня – реализовать шаблон, с которым я рассказываю =)

- ▶ плашка слева
- ▶ номера страниц и логотип
- ▶ возможность писать сноски
- ▶ работа с заголовком
- ▶ и всё это автоматизировать!

→ *примечание:* моя задача – проиллюстрировать основные идеи. Не везде предлагаемое решение будет State of the Art, где-то будет немного костылей.

← Но будет работать

|| Начнём с пустого шаблона

Будет клёвый шаблон

С кучей всего))

⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Очень продвинутый \LaTeX : программирование и работа с примитивами

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

- Простое создание команд

- Длины: единицы измерения

- Боксы и клей

- Моды и создание параграфов

- Возможности создания команд и передачи параметров в \LaTeX

Где пишем код

Код можно писать

- ▶ Прямо в начале документа
- ▶ Внутри {} (будет локализован)
- ▶ В классах
- ▶ В стилевых файлах

На уровне языка отличий класса от стиля нет.
Концептуальное различие: класс это нечто глобальное, стили могут подходить к разным классам.

Создание команд

```
\newcommand{\mycommand}[1]{something #1}
```

создаём команду

имя команды

число аргументов (может отсутствовать)

тело

↔ \renewcommand чтобы пересоздать

|| Начало .cls и .sty файлов

Класс:

```
\NeedsTeXFormat{LaTeX2e}  
\ProvidesClass{<class-name>}[<date in YYYY/MM/DD> <other  
info>]
```

Стиль:

```
\NeedsTeXFormat{LaTeX2e}  
\ProvidesPackage{<package-name>}[<date in YYYY/MM/DD> <  
other info>]
```

Специальный синтаксис

Внутри пакетов синтаксис меняется:

`\newcommand` → `\providecommand`

`\usepackage` → `\RequirePackage`

`\documentclass` → `\LoadClass`

→ происходят дополнительные проверки на отсутствие

← дублирования

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Простое создание команд

Длины: единицы измерения

Боксы и клей




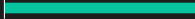
Моды и создание параграфов

Возможности создания команд и передачи параметров в \LaTeX

Длины



абсолютные значения

чаще всего используется:

pt	points	$\simeq 0.35\text{mm}$	 12pt
mm	millimeters	$\simeq 2.84\text{pt}$	 10mm
cm	centimeter	$\simeq 28.4\text{pt}, 10\text{mm}$	 1cm
in	inch	$\simeq 72.27\text{pt}, 25.4\text{mm}$	 1in

Длины

Относительные значения

em	примерно ширина буквы 'М'	 1em
ex	примерно высота буквы 'х'	 1ex

пример: если добавим команду \Huge

mm	 5mm
em	 1em
mm	 5mm
em	 1em

|| Предзаданные длины

Наиболее используемые

T _E X's	
<code>\parindent</code>	Размер отступа в параграфе
<code>\parskip</code>	Вертикальный отступ для нового параграфа
L ^A T _E X's	
<code>\textwidth</code>	Ширина текста на странице
<code>\textheight</code>	Высота текста на странице
<code>\linewidth</code>	Ширина текста в “боксе”
<code>\lineheight</code>	Высота текста в “боксе”



/Lengths#LaTeX_default_lengths man: 5.5



/16942/difference-between-textwidth-linewidth-and-hsize

`\parskip` на самом деле это клей

|| Арифметика с длинами

- ▶ Можно домножать как $0.5\text{\texttt{\textbackslash textwidth}}$
- ▶ Нельзя просто так использовать $+$, $-$, $*$, $/$
 - ▶ Это можно сделать с помощью команды
 $\text{\texttt{\textbackslash dimexpr: \textbackslash dimexpr \textbackslash textwidth - 30pt}}$

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Простое создание команд

Длины: единицы измерения

Боксы и клей

Моды и создание параграфов

Возможности создания команд и передачи параметров в \LaTeX

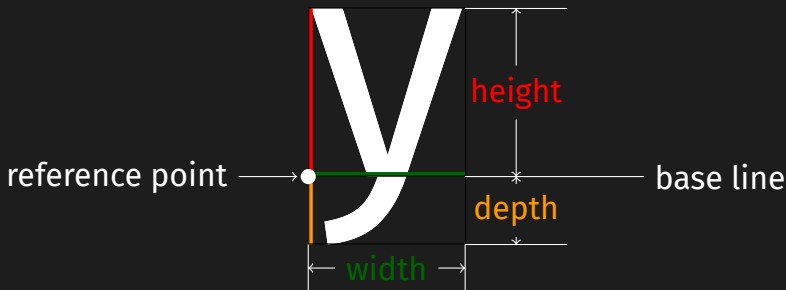
Основная идея T_EX

Символ – это **бокс**
он – часть слова, которое **бокс**
слова соединены **клеем**
в предложения и параграфы.

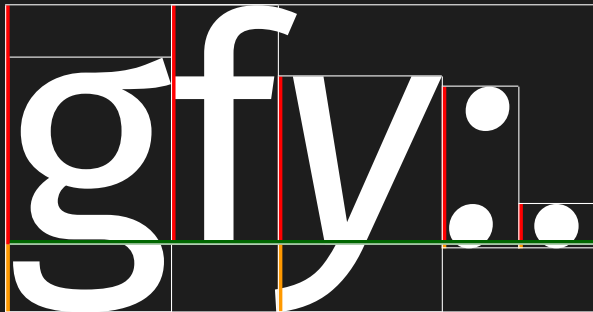
Параграф, кстати, это **бокс**
он соединён с другими **клеем**
в страницу. Которая – **бокс**

таблица, картинки, ... – это **бокс**

Параметры бокса: высота, глубина, ширина



Как T_EX объединяет боксы



- Обратите внимание, что бокс не совпадает полностью с буквой. “f” выходит за неё, точка
- ← занимает не всё пространство бокса

Два типа боксов

Есть два типа боксов:

- ▶ “Горизонтальный” бокс – стыкуется к другим горизонтальным боксам. Его параметр – это ширина.
- ▶ “Вертикальный” бокс – стыкуется с другими вертикальными боксами. Его параметры – высота и глубина.

T_EX: Горизонтальные и вертикальные боксы

```
\hbox to 20pt {hello world}
```

“горизонтальный” бокс

какой длины его считает T_EX (может отсутствовать)

тело бокса

```
\vbox to 20pt {hello world}
```

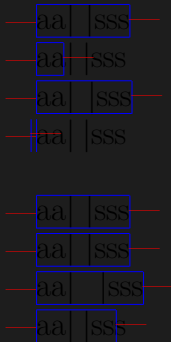
“вертикальный” бокс

какой высоты его считает T_EX (может отсутствовать)

тело бокса

Горизонтальный бокс с разными параметрами

```
\boxing{\hbox{aa|sss}}\\
\boxing{\hbox to 20pt{aa|sss}}\\
\boxing{\hbox to 70pt{aa|sss}}\\
\boxing{\hbox to -4pt{aa|sss}}\\
\boxing{\hbox{aa|sss}}\\
\boxing{\hbox spread 0pt{aa|sss}}\\
\boxing{\hbox spread 10pt{aa|sss}}\\
\boxing{\hbox spread -10pt{aa|sss}}
```





“Горизонтальный” бокс

Использование

`\hbox to -1pt{/}=` \neq

```
\begin{tabbing}
\hbox to 4em{} \= \hbox to 4em{}\kill
a \> b\\
hello\> world!      hello      world!
\end{tabbing}
```


Вертикальный бокс с разными параметрами

```

\boxing{\vbox{a}}
\boxing{\vbox{yf}}
\boxing{\vbox to 15pt{yf}}
\boxing{\vbox to 5pt{yf}}
\boxing{\vbox{yf}}
\boxing{\vbox to -10pt{yf}}
\boxing{\vbox spread 0pt{yf}}
\boxing{\vbox spread 30pt{yf}}
\boxing{\vbox{yf}}
\boxing{\vtop{yf}}
\boxing{\vtop spread -40pt{yf}}
\boxing{\vtop to 30pt{yf}}

```

\overline{a} \overline{yf} \overline{yf} \overline{yf}
 \overline{yf} \overline{yf} \overline{yf}
 \overline{yf} \overline{yf} \overline{yf}



Двигаем буквы

```
Write as L\raise0.5ex\hbox
           {A}T\lower0.5ex\hbox{E
           }X
\vspace{3ex}
\ vbox{L}
\moveleft0.5em\ vbox{A}
\ vbox{T}
\ moveright0.5em\ vbox{E}
\ vbox{X}
```

Write as L^AT_EX

L
A
T
E
X

Можно использовать команды `\raise`, `\lower`,
`\moveleft`, `\moveright`

Л^AT_EX: Горизонтальные боксы

```
\mbox{hello world}
```

“горизонтальный” бокс

тело бокса

```
\makebox[20mm][c]{hello world}
```

“горизонтальный” бокс

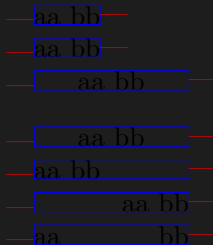
какой длины бокс

выравнивание внутри бокса

тело бокса

Горизонтальный бокс

```
\boxing{\mbox{aa bb}}  
\boxing{\makebox{aa bb}}  
\boxing{\makebox[20mm]{aa  
bb}}  
\boxing{\makebox[20mm][c]{  
aa bb}}  
\boxing{\makebox[20mm][l]{  
aa bb}}  
\boxing{\makebox[20mm][r]{  
aa bb}}  
\boxing{\makebox[20mm][s]{  
aa bb}}
```



Л^AT_EX: Вертикальные боксы

```
\parbox [c] [20pt] {100pt} {hello world}
```

“вертикальный” бокс

выравнивание внутри бокса

какой высоты бокс

какая ширина бокса

тело бокса

```
\raisebox {20pt} [10pt] [50pt] {hello world}
```

“подъёмный” вертикальный бокс

насколько поднимаем

высота бокса

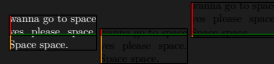
глубина бокса

тело бокса

Вертикальные боксы

```
Space space \parbox{8em}{  
  wanna go to space yes  
  please space. Space  
  space.} Go to space.  
\boxingDim{\parbox{8em}{  
  wanna go to space yes  
  please space. Space  
  space.}}  
\boxingDim{\parbox[t]{8em  
  }{wanna go to space  
  yes please space.  
  Space space.}}  
\boxingDim{\parbox[b]{8em  
  }{wanna go to space  
  yes please space.  
  Space space.}}
```

wanna go to space
Space space yes please space. Go to space.
Space space.



Пробелы

Клей и Керны предоставляют пробелы между боксами.

```
G\hskip0em lu\hskip0.5em e  
and k\kern0em e\kern  
0.5em rn provides...
```

Glu e and ke rn provides...

Что такое клей

Но клей – это больше, чем просто “пробел” между боксами.

Клей это **растяжимый** пробел между боксами.

`\hskip 2em plus 0.5em minus 0.6em`
добавить горизонтальный клей (TeX). `\vskip` добавит вертикальный
базовый отступ
насколько может растянуться (опционно)
насколько может сжаться (опционно)

`\hspace{2em plus 0.5em minus 0.6em}`
добавить горизонтальный клей (LaTeX). `\vspace` добавит вертикальный.

Где клей добавляется неявно

Между словами и предложениями. Тут много клея.

Между словами и предложениями. Тут много клея.

Между словами и предложениями. Тут много клея.

Между словами и предложениями. Тут много клея.

Между словами и предложениями. Тут много клея.

Между словами и предложениями. Тут много клея.

Между словами и предложениями. Тут много клея.

→ P.S. Тут \hbox растянутый в диапазоне -10pt—40pt.

Заметьте, что между предложениями растяжение больше, чем между словами. И что дефолтное растяжение – третья строка – не минимально.

← А ещё клей есть между параграфами.

Бесконечный клей

```
\hbox to 50mm{\hskip0em plus 1fil\  
  relax 1fil and 1fil \hskip0em  
  plus 1fil\relax}  
\hbox to 50mm{\hskip0em plus 1fil\  
  relax 1fil and 2fil \hskip0em  
  plus 2fil\relax}  
\hbox to 50mm{\hskip0em plus 1fill\  
  relax fill and fill \hskip0em  
  plus 1fill\relax}  
\hbox to 50mm{\hskip0em plus 1filll\  
  relax fill vs fil \hskip0em  
  plus 999fil\relax}  
\hbox to 50mm{\hskip0em plus 1fillll\  
  relax fillll and fillll \hskip0  
  em plus 1fillll\relax}  
\hbox to 50mm{\hskip0em plus 1fillll\  
  relax fillll vs fill \hskip0em  
  plus 999fill\relax}
```

1fil and 1fil
1fil and 2fil
fill and fill
fill vs fil
filll and filll
filll vs fill

fil, fill, filll добавляют бесконечность разной
“степени”

Аббревиатуры

Можно использовать:

`\hfil` `\hfill` `\hspace{\fil}` `\hspace{\fill}`

`\vfil` `\vfill` `\vspace{\fil}` `\vspace{\fill}`

`\hss`, `\vss` – бесконечный клей как в *plus* так и в *minus*.

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Простое создание команд

Длины: единицы измерения

Боксы и клей

Моды и создание параграфов

Возможности создания команд и передачи параметров в \LaTeX



Моды

\TeX имеет 3(6) мод:

1. **Vertical mode.** [Создание главного вертикального списка, из которого получаются страницы.]
2. **Internal vertical mode.** [Вертикальный список для vbox.]
3. **Horizontal mode.** [Создание горизонтального списка для параграфов.]
4. **Restricted horizontal mode.** [Создание горизонтального списка для hbox.]
5. **Math mode.** [Создание математической формулы внутри горизонтального списка.]
6. **Display math mode.** [Создание математической формулы и положение её на отдельную строку, прерывание параграфа.]



Разница между модами

Много мелких различий. Например:

- ▶ в горизонтальной моде только первый пробел имеет значение
- ▶ в математической моде шрифт по умолчанию - италик, пробелы игнорируются
- ▶ в выделенной математической моде операторы рисуются больше, чем в обычной
- ▶ в вертикальной моде все пробелы и `<return>` игнорируются

|| Ещё чуть-чуть о математической моде

В реальности у нас есть 4 стиля:

Display style	<code>\displaystyle</code>	A	главный стиль для выделенной формулы
Text style	<code>\textstyle</code>	A	главный стиль для внутритекстовой формулы
Script style	<code>\scriptstyle</code>	A	главный стиль для индексов
Script-script style	<code>\scriptscriptstyle</code>	A	главный стиль для под-индексов

Создание параграфов

Для перфекционистов

Microsoft Word 2008

Call me Ishmael. Some years ago – never mind how long precisely – having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin

Adobe InDesign CS4

Call me Ishmael. Some years ago – never mind how long precisely – having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin

pdf-L^AT_EX 3.1415926

Call me Ishmael. Some years ago – never mind how long precisely – having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin

Hyphenation and inter-word spacing statistics

	Word	InDesign	pdf-L ^A T _E X
Number of hyphenations	9	10	4
SD of IWS (pt)	2.26	1.94	1.42
Maximum IWS (pt)	14.4	13.2	9.0
Number of lines with IWS > 9 pt	5	2	0

SD: standard deviation; IWS: inter-word spacing





Создание параграфов

“это, по факту, наверно
самый интересный
аспект всей системы T_EX”
D. Knuth, the T_EXBook

|| Обзор

- ▶ Каждый параграф создаётся целиком: слова в конце параграфа могут повлиять на расположение строк в начале.
- ▶ T_EX никогда не положит слова ближе, чем позволяет клей.
- ▶ T_EX пробует все комбинации разрывов строк. Для каждого варианта и каждой строки T_EX вычисляет параметр *badness*. Если он меньше `\tolerance`, T_EX попытается создать параграф с минимальным числом переносов слов.
- ▶ если у T_EX'а не получается, он выдаст **Overfull** или **Underfull** ворнинги.

|| Как управлять переносами слов

Локально: использовать \-

as in this ve\ -ry long se\ -nta\ -nce

Глобально:

\hyphenation{some-thing poss-ible}

\- это короткая версия для команды

\discretionary{hpre-break texti}{hpost-break texti}{hno-break texti}

так же: Т_EX никогда не перенесёт слово с “/”. используйте вместо этого \slash, если перенос нужен. А \uchyph=0 запретит перенос слов с Большой буквы.

|| Управление разрывом строк вручную

Никогда не прерываются: неразрывный пробел~,
`\nobreak`, `\nolinebreak`

Всегда: `\`, `\break`, `\linebreak`

Ещё можно использовать `\obeylines` И тогда разрывы будут происходить там же, где в исходном коде.

кстати: `\` имеет опционный параметр: вертикальный отступ после себя. А `\smallskipamount`, `\medskipamount` и `\bigskipamount` отвечают за разрыв после параграфа. `\linebreak` имеет опционный параметр: [0-4] насколько сильно ваше желание прервать строку после этого.

|| Алгоритм: часть 1

1. Т_ΕХ создаёт варианты без переноса слов.
Проверяет параметр *badness* с параметром `\pretolerance`.
2. $badness \simeq 100 \cdot \text{proportion-between-the-normal-glue-and-its-stretching/compression}$ ³
3. если проверка с `\pretolerance` провалилась, Т_ΕХ попробует все комбинации разрывов строк, чтобы *badness* была меньше, чем `\tolerance`

|| Алгоритм: часть 2

1. разрывы строк допустимы лишь в следующих местах:
 - 1.1 клей
 - 1.2 керн, после которого идёт клей
 - 1.3 математика (\$) с последующим клеем
 - 1.4 ручное или автоматическое прохождение штрафа
 - 1.5 discretionary разрыва
2. Пенальти для клея – 0. Для разрыва это `\hyphenpenalty=` или `\exhyphenpenalty=`. Пенальти можно добавить вручную через `\penalty`
3. Пенальти может быть как положительная, так и отрицательная. Если оно $> 10^4$ тут никогда не будет разрыва, а если $< -10^4$ разрывы будут всегда

умолчания: `\hyphenpenalty=50`, `\exhyphenpenalty=50`
а ещё можно добавлять доп пространства между текстом и математикой с `\mathsurround`

|| Алгоритм: часть 3

1. В реальности, \TeX пытается минимизировать *demerits*. Он пропорциональен `badnesses`, `\linepenalty` (определяет, насколько сильно вы просите \TeX уменьшить число строк) и `penalty`
2. \TeX так же берёт в расчёт и добавляет штрафы, если две строки подряд имеют перенос слов (`\doublehyphendemerits`), и если строки *визуально несовместимы* (например: растянутая линия с ужатой) (`\adjdemerits`) и если предпоследняя строка абзаца заканчивается *discretionary* (`\finalhyphendemerits`)

Умолчания: `\linepenalty=10`, `\adjdemerits=10000`, `\doublehyphendemerits=10000`,
`\finalhyphendemerits=5000`.

`\hfuzz=...` добавит максимальную длину выравнивания строки

|| Что ещё

- ▶ Используйте `\narrow` чтобы сузить строки
- ▶ `\looseness=-1` чтобы попросить \TeX попробовать уменьшить число строк в параграфе
- ▶ `\prevgraf` показывает текущую строку параграфа
- ▶ `\vadjust` добавляет что-то в вертикальный список после каждого параграфа. Например, так мы добавили звёздочку слева
- ▶ `\everypar` добавляет что-то после каждого параграфа
- ▶ `\parfillskip` — клей после каждой строки
- ▶ `\parskip` — вертикальный клей между параграфами

|| Создание страниц

- ▶ Кнут создал T_EX тогда, когда памяти оптимизировать всю страницу ещё не хватало.
- ▶ T_EX ищет лучший разрыв для текущей страницы и удаляет страницу из памяти.
- ▶ В целом алгоритм примерно такой же.
- ▶ Можно использовать `\penalty` or `\nobreak` в вертикальном моде
- ▶ Можно использовать чтобы убрать привязку вертикальную к низу страницы
- ▶ По аналогии с параграфами, можно использовать `\newpage`, `\pagebreak`, `\nopagebreak`

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Простое создание команд

Длины: единицы измерения

Боксы и клей

Моды и создание параграфов

Возможности создания команд и передачи параметров в \LaTeX



Оptionные аргументы в создании команд

```
\newcommand{\test}[2][my default arg] имеет{ умолчание #1, задаётся #2}
```

общее число аргументов
какие аргументы по умолчанию
(по порядку с начала)

```
\usepackage{xargs}
```

```
\newcommandx {\ testOpt }[3][3= def opt val]
```

Для именованных команд можно использовать пакет *keyval*
или *pgfkeys*

Передача параметров в пакет

`\RequirePackage{kvoptions}` – используем пакет
`\SetupKeyvalOptions{family=KVCSC, prefix=KVCSC@}` – задаём namespace

`\DeclareStringOption[noarg]{argname}[default arg value]`
значение, если аргумент не передан совсем `\usepackage{mypackage}`
имя аргумента для передачи, может быть использована как
`\usepackage[myarg=smth]{mypackage}`
дефолтный аргумент в случае, если пакет вызывает-
ся как `\usepackage[smth]{mypackage}`

`\ProcessKeyvalOptions*` – непосредственно подставляем полученные значения

`\KVCSC@argname` – под таким именем аргумент будет доступен в командах

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Очень продвинутый \LaTeX : программирование и работа с примитивами

Три взгляда на область

→ Нас ждёт манипулирование примитивами: длинами, боксами, клеем, счётчиками,.. Условные операторы, циклы и рекурсия, работа с файловой системой. В работе с этими вещами есть три подхода (Мы выделяли их и раньше, но теперь пора обратить
← внимание):

1. **TeX- подход.** Использование тех команд и макросов, которые вшиты в TeX, созданный Доннальдом Кнутом
2. **LaTeX- подход.** Использование расширения TeX'a, созданное Лесли Лэмпортом
3. **packageLaTeX- подход.** Использование команд из расширений LaTeX'a разной степени стандартности


Особое внимание будем уделять первому подходу как базовому.

Соглашение о наименовании

- ▶ Если команда сделана для пользователей, используется короткое имя и lowercase: `\section`, `\emph` and `\times`
- ▶ Если команда для создателей других пакетов, используется CamelCase: `\InputIfFileExists` `\RequirePackage` `\PassOptionsToClass`
- ▶ Для “приватных” команд используется @. Большая часть внутренних команд \TeX а использует эту букву внутри: `\@tempcnta`, `\@ifnextchar`, `\@eha`.
Чтобы использовать такие команды внутри файлов .tex, это использование надо окружить `\makeatletter`, `<use command>`, `\makeatother`

TeX это Тьюринг полный язык программирования

[https://stackoverflow.com/questions/2968411/
ive-heard-that-latex-is-turing-complete-are-there-any-programs-written-in-latex](https://stackoverflow.com/questions/2968411/ive-heard-that-latex-is-turing-complete-are-there-any-programs-written-in-latex)

 [/Articles/LaTeX_is_More_Powerful_than_you_Think_-_Computing_the_Fibonacci_Numbers_and_Turing_Completeness](https://www.researchgate.net/publication/220121226_LaTeX_is_More_Powerful_than_you_Think_-_Computing_the_Fibonacci_Numbers_and_Turing_Completeness)
<http://sdh33b.blogspot.com/2008/07/icfp-contest-2008.html>

Итак, сегодня...

Очень продвинутый `TeX`: программирование и работа с примитивами

Создание команд

Условные операторы

Работа с примитивами

Циклы и рекурсия

Работа с файловой системой

Манипулирование с именами команд и ещё
несколько ключевых слов

Дебаггинг и логгирование

Создание команд

В T_EX для создания команд используется `\def`.

```
\def\lookAtMe#1{\vbox{I'm
  mister #1 look at me!}}
\def\dfdx#1#2{\ensuremath{\frac{\partial}{\partial #1}\partial #2}}}
\lookAtMe{Gosha} \lookAtMe{
  Misha} \lookAtMe{Tema}
\dfdx{g}{y}
$$\dfdx{v}{z} = 5x$$
```

I'm mister Gosha look at me!
 I'm mister Misha look at me!
 I'm mister Tema look at me!
 $\frac{\partial g}{\partial y}$

$$\frac{\partial v}{\partial z} = 5x$$

→ Используйте префикс `\global`, чтобы макрос был доступен вне стандартной области видимости.

И `\long` чтобы его частями могли стать несколько

← параграфов.

Pattern matching

Зачем нужно писать `\def\name#1#2#3...` ВМЕСТО чего-то
наподобие \TeX , `\def\name[5]`

Ради таких штук:

```
\def\parseLine #1, #2 \par {arg1: #1, arg2: #2}
```

начинает собирать первый аргумент
 первый аргумент – всё до ближайшей запятой с пробелом
 начинает собирать второй аргумент
 всё до нового параграфа
 что делать с аргументами

`\parseLine Hello , World` в первом аргументе будет “Hello”, во
втором – “World”

Итак, сегодня...

Очень продвинутый `TeX`: программирование и работа с примитивами

Создание команд

Условные операторы

Работа с примитивами

Циклы и рекурсия

Работа с файловой системой

Манипулирование с именами команд и ещё несколько ключевых слов

Дебаггинг и логгирование

Сравнить строки(макросы)

```

\def\ttest#1#2{
\def\a{#1}
\def\b{#2}
\ifx\a\b
    yes
\else
    no
\fi}

\ttest{ab}{ab} \ttest{ba}{ab}

```

yes no

```

\ifx<first>\<second> <code1> [\else <code2>] \fi

```

Сравнить числа

```
\ifnum \year > 2022 как дела, потомки? \else свеженькая лекция! \fi
```

условие

что происходит при прохождении условия?

опциональный блок *else*

что при провале условия

Кстати, свеженькая лекция!

“именованные” условия: `\newif`

```
\newif\ifIAmMad
```

```
\IAmMadtrue
```

I am mad

```
\ifIAmMad
```

```
I am mad
```

```
\else
```

```
I am ok
```

```
\fi
```

→ Т.е. сначала мы объявляем, что хотим использовать условие под таким вот именем. Потом – определяем, сейчас это true или false.

И где-то, где уже реализуется логика, код выполняется в зависимости от того, считается ли *В данный момент* условие истинным или ложным. Что мы задали раньше.

|| Проверка мод

- ▶ `\ifmmode` это математическая мода?
- ▶ `\ifvmode` это вертикальная мода?
- ▶ `\ifhmode` это горизонтальная мода?
- ▶ `\ifinner` это одна из следующих мод: internal vertical mode, restricted horizontal mode, (nondisplay) mathmode?

остальные “if”ы могут быть найдены в [kn: 20](#). Например `\ifdim` проверяет длины, `\ifeof` – конец файла,

`\ifcsname` – существование команды по имени

Ещё полезна инверсия бывает:

```
\def\ifnot#1{#1\else\expandafter\expandafter\fi\iffalse\iftrue\fi}
```



Условия в L^AT_EX: `\ifthenelse`

```
\usepackage{ifthen}
```

```
\ifthenelse {\equal{aa}{bb}}{world is mad}{world is good}
```

проверка условия для строк. Для чисел можно `>`, `<`, ..

что делать если условие прошло

что делать если условие
провалилось

Ещё есть хороший пакет `xstring`. Там проверка строк, подстрок, операции с ними...

А проверить pdfLaTeX, XeLaTeX и LuaTeX можно с

Итак, сегодня...

Очень продвинутый **TeX**: программирование и работа с примитивами

Создание команд

Условные операторы

Работа с примитивами

Циклы и рекурсия

Работа с файловой системой

Манипулирование с именами команд и ещё
несколько ключевых слов

Дебаггинг и логгирование

Счётчики – counters

Счётчик – просто целое число. Но используется он в огромном числе вариантов.

Номер раздела, слайда, уравнения, цитаты, нумерованный список и многое другое.

Например

- ▶ номер этого слайда - `\insertframenumbers=74`
- ▶ номер страницы при этом – `\thesection=75`
`\thepage=75`

Определение и манипулирование счётчиками

- ▶ `\newcounter{abcd}` – объявить счётчик
- ▶ `\setcounter{abcd}{2022}` – присвоить счётчику значение
- ▶ `\addtocounter{abcd}{-42}` – добавить число

|| Отображение счётчиков

<code>\arabic{countname}</code>	1	2	3	4	5	6	7	8	9
<code>\alph{countname}</code>	a	b	c	d	e	f	g	h	i
<code>\Alph{countname}</code>	A	B	C	D	E	F	G	H	I
<code>\roman{countname}</code>	i	ii	iii	iv	v	vi	vii	viii	ix
<code>\Roman{countname}</code>	I	II	III	IV	V	VI	VII	VIII	IX
<code>\fnsymbol{countname}</code>	*	†	‡	§	¶		**	††	‡‡



Доминирование счётчиков

“Доминирование” одного счётчика над другим – значит при обновлении первого счётчика второй вернётся к 1.

Доминирование – механизм, благодаря которому все подразделы могут иметь нумерацию, привязанную к разделу, а не сквозную.

Добавить доминирование счётчику



`\newcounter{task}`



`\newcounter{task}[section]`

`\newcounter{<slave>}[<master>]` обнулит
значение <slave> при изменении <master>

`\addtocounter{task}{1}`



`\refstepcounter{task}`

Используйте `\refstepcounter{<counter>}`, чтобы
правильно работал механизм ссылок



`\renewcommand{\thetask}{\arabic{section}.\arabic{task}}`

Переопределите `\renewcommand{\the<counter>}`,
чтобы правильно работали ссылки `\ref`

Счётчики в T_EX

- ▶ **Определить свой** `\newcount\<countname>` как `\newcount\mycounter`
- ▶ **Задать значение** `\<countname>=<number>` Или используйте `\countdef`. Как `\countdef\mynumber=43`
- ▶ **Добавить число** `\advance\<countname> by <number>`. Так же можно использовать `\multiply` и `\divide`. А ещё `\numexpr`.
- ▶ **Показать значение** `\the\<countname>` или `\number` или `\romannumeral`

Длины – lenghts

Длина – число + размерность. В реальности кратное
“scaled points” = $1/2^{16}$ pt.

Длины тоже можно прибавлять, умножать, делить...
Причём размерность приведётся.

Манипулирование длиной в L^AT_EX

- ▶ **Определить длину** `\newlength{\<lenname>}`
- ▶ **Задать длину** `\setlength`
- ▶ **Добавить что-то к длине** `\addtolength`
- ▶ **Отобразить длину** `\the\<lenname>`. Но ещё можно использовать `\usepackage{printlen}` и потом `\uselengthunit, \printlength`

Манипулирование длиной в T_EX

- ▶ **Определить длину** `\newdimen<lenname>`
- ▶ **Задать длину** `<lenname>=<len>`
- ▶ **Добавить к длине** `\advance<lenname> by <len>`. Ещё есть `\multiply` и `\divide`. И `\dimexpr`.
- ▶ **Показать длину** `\the<lenname>`

Боксы – boxes

Боксы – прямоугольники, по которым \TeX определяет положение объектов.

Боксы можно сохранять в память (без печати), узнавать ширину-высоту-глубину, печатать



Манипулирование боксом в L^AT_EX

- ▶ **Определить бокс** `\newsavebox{\<boxname>}`
- ▶ **Задать бокс** `\savebox`
(`\savebox{\mybox}{\hbox{LaTeX content}}`)
- ▶ **Напечатать содержимое не удаляя бокс:** `\usebox`
(`\usebox{\mybox}`)
- ▶ **Узнать размеры:**
 1. Создать переменную длины: `\newlength`
 2. Определить переменную одним из измерений боксов:
 - ▶ ширина:
`\settowidth{\<len-var>}{\usebox{\<box>}}`
 - ▶ высота:
`\settoheight{\<len-var>}{\usebox{\<box>}}`
 - ▶ глубина:
`\settodepth{\<len-var>}{\usebox{\<box>}}`



Манипулирование боксом в T_EX

- ▶ **Определить бокс** `\newbox<boxname>`
- ▶ **Задать бокс** `\setbox<boxname>=<box>`
(`\setbox\mybox=\hbox{LaTeX content}`)
- ▶ **Напечатать содержимое не удаляя бокс:**
`\copy<boxname>`
- ▶ **Напечатать содержимое и удалить бокс из памяти:** `\box<boxname>`
- ▶ **Узнать размеры:** ширина: `\wd`, высота: `\ht`,
глубина: `\dp` (`\dp\mybox`)



Остальные примитивы T_EX

В дополнение к длинам, боксам и счётчикам в T_EX есть:

- ▶ `\skip`, `\muskip` – регистры для клея в обычной и математической моде соответственно
- ▶ `\toks` – регистры для строк

Итак, сегодня...

Очень продвинутый `TeX`: программирование и работа с примитивами

Создание команд

Условные операторы

Работа с примитивами

Циклы и рекурсия

Работа с файловой системой

Манипулирование с именами команд и ещё несколько ключевых слов

Дебаггинг и логгирование



Циклы и рекурсия в T_EX

Циклы:

```
\newcount\icount
\icount=10
\loop A?
\ifnum\icount>0
B! \advance \icount by -1
\repeat
```

A? B! A? B! A? B! A? B!
 A? B! A? B! A? B! A? B!
 A? B! A? B! A?

→ `\loop` для старта, `<code>` внутри, потом один из операторов серии `\if<...>`, ещё код `<code>`, заканчиваем `\repeat` (вместо `\fi`).

рекурсия:

```
\def\requ#1{\ifnum#1>0 A?
\requ{\numexpr#1 - 1}
}\fi}
\requ{8}
```

A? A? A? A? A? A? A?
 A?

kn: 20.

`\loop` это просто `\def\loop#1\repeat{\def\body{#1}\iterate},`
 а `\iterate: \def\iterate{\body\let\next=\iterate\else\let`
`\next=\relax\fi\next}`



Циклы в L^AT_EX

```
\usepackage{forloop}
\newcounter{themenumber}
\forloop{themenumber}{1}{\
  value{themenumber} < 5}{
A?
}
```

A? A? A? A?

```
\usepackage{forloop}
```



```
\usepackage{pgffor}
\foreach \n in {0,...,4}{\n\
  space}

\foreach \n in {apples,
  burgers, cake}{Let's eat
\n.\par}
```

0 1 2 3 4

Let's eat apples.

Let's eat burgers.

Let's eat cake.

← \usepackage{pgffor}, часть пакета pgf, часть TikZ

Итак, сегодня...

Очень продвинутый `TeX`: программирование и работа с примитивами

Создание команд

Условные операторы

Работа с примитивами

Циклы и рекурсия

Работа с файловой системой

Манипулирование с именами команд и ещё
несколько ключевых слов

Дебаггинг и логгирование

Запись в ФС

- ▶ `\newwrite` объявляет переменную:
`\newwrite\myfile`.
 - ▶ `\openout` открывает файл на запись:
`\openout\myfile=outfile.txt`.
 - ▶ Можно использовать `\jobname` в качестве имени файла.
 - ▶ `\write<register>` собственно производит запись:
`\write\myfile{{\noexpand\bf Hello}\World}`
- ▶ `\noexpand` тут не даёт команде запуститься, а
← просто записывает её (см также `\expandafter`)
- ▶ `\closeout` закрывает файл на запись:
`\closeout\myfile`

Чтение из ФС

- ▶ `\newread` объявляет переменную:
`\newread\myfile`.
- ▶ `\openin` открывает файл на чтение:
`\openin\myfile=infile.txt`.
- ▶ `\read<register> to\<newvariable>` собственно читает в переменную: `\read\myfile to\myline`
- ▶ `\ifeof` проверяет, достигли ли мы конца файла:
`\ifeof\myfile`
- ▶ `\closein` закрывает файл на чтение:
`\closein\myfile`



Работа с командной строкой

```
\immediate\write18{wget https://<url>.png -O image.png}
```

запустить команду сразу как увидим (иначе запуск при формировании страницы)
запись в 18 регистр = в командную строку
текст команды

- При компиляции из командной строки надо использовать
- ← ключи `--enable-write18 -interaction=nonstopmode`

Л_AT_EX и предсказание будущего

Т_EX читает код последовательно. Т_EX знает лишь о той части кода, которая уже прочитана на момент отрисовки данной страницы.

Все перекрёстные ссылки, любая информация с “будущих страниц” требует запись этой информации в файл и чтение при новом запуске.

Вот почему Л_AT_EX так часто надо запускать 2 раза.

Итак, сегодня...

Очень продвинутый `TeX`: программирование и работа с примитивами

Создание команд

Условные операторы

Работа с примитивами

Циклы и рекурсия

Работа с файловой системой

Манипулирование с именами команд и ещё
несколько ключевых слов

Дебаггинг и логгирование



\let


`\let` Позволяет копировать описание макроса в новое имя. После этого старое можно переназначить:



- ▶ `\def\a{hello} \let\b=\a` назначит `\b` слово “hello”.
- ▶ `\def\a{\b\ world} \a` выведет “hello world”



Expansion

- ▶ `\expandafter` говорит, что сначала должно выполниться содержимое команды, а лишь потом сама команда.
 - ▶ `\def\aaa{\uppercase{\a, my oborona}}` вернёт “aaa, MY OBORONA”
 - ▶ `\def\aaa{\uppercase\expandafter{\a, my oborona}}` вернёт “AAA, MY OBORONA”
- ▶ `\edef` полностью раскроет все макросы внутри
 - ▶ `\def\testp{\thepage}` всегда будет возвращать текущую страницу
 - ▶ `\edef\testp{\thepage}` всегда будет возвращать страницу, на которой объявили макрос

 [Articles/How_does_%5Cexpandafter_work%3A_The_meaning_of_expansion](#)

 [Articles/How_does_%5Cexpandafter_work%3A_From_basic_principles_to_exploring_TeX%27s_source_code](#)  [451/when-to-use-edef-noexpand-and-expandafter](#)

Манипулирование именами команд



- ▶ `\csname textit\endcsname` прям создаст команду наклонного шрифта:
`\csname textit\endcsname{my text}` даст *my text*
- ▶ `\string\textit` вернёт просто `\textit`



Catcodes

что отвечает за комментарий, а что – символ команды?

```
{
\catcode'\ [=1 \catcode
'\ ]=2
\catcode'\ {=12 \catcode
'\ }=12
\catcode'\ #=12
\catcode'\ %=12 \catcode
'\ ?=14
a#b % a comment?? yes

[{\aasd}]
]
```

a#b % a comment
{aasd}

catcodes отвечают, какой символ будет комментарием, с какого – начинается команда, какими обозначаются группы {},...

Итак, сегодня...

Очень продвинутый `TeX`: программирование и работа с примитивами

Создание команд

Условные операторы

Работа с примитивами

Циклы и рекурсия

Работа с файловой системой

Манипулирование с именами команд и ещё
несколько ключевых слов

Дебаггинг и логгирование

Способы дебаггинга

В TeX можно вывести интересующую информацию в лог-файл. Есть три способа это сделать:

- ▶ **show-команды** – выводят в лог раскрытие конкретного макроса или содержимое конкретного примитива
- ▶ **tracing-команды** – меняют уровень логгирования, позволяя заглянуть внутрь всей работы
- ▶ **\message** – просто что-то вывести в лог



Show-команды

Использование: `\show\macros`

`\show` логирует, что внутри макроса

`\showthe` логирует содержимое счётчика или длины

→ `\showbox` логирует содержимое бокса (какой там клей, какие слова, etc)

`\showboxdepth` показывает глубину
вложенности бокса

`\showboxbreadth` сколько элементов на этом
уровне

`\showlists` описывает контент списка боксов во всех 4х
нематематических режимах

`\showhyphens{Word}` показывает возможные переносы
слова Word.





tracing-команды

Использование: `\tracingmacros=1` <что-то>
`\tracingmacros=0`

Если >0, пишет в лог-файл...

`\tracingcommands` команды

`\tracingmacros` раскрытие макросов и их аргументов

`\tracingpages` стоимость расчёта страницы

→ `\tracinglostchars` каких символов нет в текущем шрифте

`\tracingonline` пишет и в терминал и в лог

`\tracingoutput` содержимое боксов

`\tracingparagraphs` сводку по расчёту переноса строк

`\tracingrestores` save-stack

`\tracingstats` статистику использования памяти

← `\tracingall` абсолютно всё из вышеперечисленного

`\message` и `\typeout`

Пишет сообщение в лог:

- ▶ `\message{<msg>}` – $\text{T}_\text{E}\text{X}$ -command
- ▶ `\typeout{<msg>}` – \LaTeX -command

|| P.S. нотация шрифтов

Иногда вы можете увидеть подобный ворнинг в лог-файле: **LaTeX Font Warning: Font shape 'T1/calligra/bx/n' undefined.** Как читать такое обозначение?

T1	calligra	bx	n	
OT1	cmr	m	it	10
encoding	font family	series	shape	font size

Подробнее см. “fntguide”.

Что сегодня узнали

1. Прimitives в \LaTeX

- 1.1 Как \TeX видит наш документ: боксы и клей
- 1.2 Какие primitives существуют: длины, счётчики и другое
- 1.3 Как манипулировать primitives

2. Программирование в \TeX и \LaTeX





- 2.1 Создание макросов
- 2.2 Условные операторы
- 2.3 Циклы и рекурсия
- 2.4 Операции ввода-вывода
- 2.5 Отладка

Что сегодня сделали

- ▶ Создали шаблон с нуля
- ▶ Научились создавать свои команды и работать с параметрами шаблонов
- ▶ Сделали красивый футер: расположили заметку по центру, выставили пробелы, научились расширять пространство для заметки
- ▶ Построили прогрессбар, привязанной к секции

Ссылки на литературу

color from the footnotes corresponds to references' color.

- ▶ kn: Д. Кнут “The \TeX Book”
- ▶ lv: Львовский “Набор и вёрстка в системе \LaTeX ”
- ▶ man: “ \LaTeX 2 ϵ : An unofficial reference manual” так же на сайте <https://latexref.xyz/>
- ▶ : <https://tex.stackexchange.com/questions>
- ▶ : <https://en.wikibooks.org/wiki/LaTeX>
- ▶ : <https://www.overleaf.com/learn/latex>
- ▶ : <https://www.tug.org/utilities/plain/cseq.html>

|| Распространение

- ▶ the pdf-version of the presentation and all printed materials can be distributed under license Creative Commons Attribution-ShareAlike 4.0
<https://creativecommons.org/licenses/by-sa/4.0/>
- ▶ The source code of the presentation is available on <https://github.com/Lavton/latexLectures> and can be distributed under the MIT license https://en.wikipedia.org/wiki/MIT_License#License_terms