

Продвинутый L^AT_EX

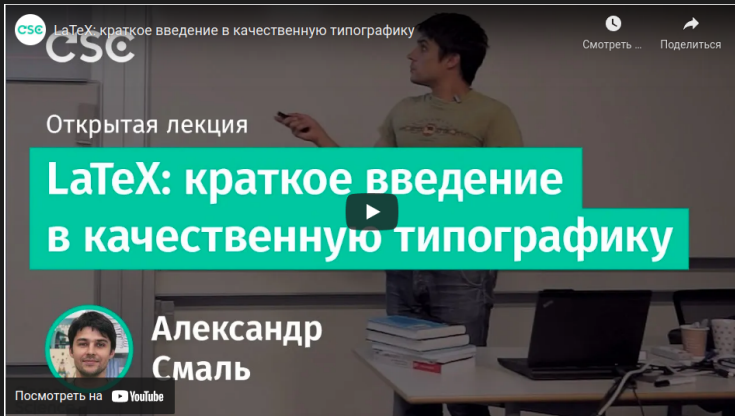
Типография и работа с командами

Антон Лиознов

CSC

2022

Предыстория



TeX vs Power Point

- ▶ Для большинства задач – дело вкуса
- ▶ Написание в TeX обычно медленнее
- ▶ В TeX гораздо проще делать “по правилам”, единообразно
- ▶ В TeX есть больше свободы в создании необычных фиш. Хотя это свобода редко нужна.
 - ▶ Но по ходу лекции вы, надеюсь, будете замечать вещи, которые было бы проблематично сделать в Power Point

\LaTeX vs \TeX

1. \TeX - подход. Использование тех команд и макросов, которые вшиты в \TeX , созданный Дональдом Кнутом начиная с 1978 г
2. \LaTeX - подход. Использование расширения \TeX 'а, созданное Лесли Лэмпортом. (версия 2 ϵ появилась в 1994 г)
3. `package` \LaTeX - подход. Использование команд из расширений \LaTeX 'а разной степени стандартности

Особое внимание будем уделять первому подходу как базовому.

Что сегодня узнаем?

1. Прimitives в \LaTeX

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

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 узнаем?

1. Примитивы в \LaTeX

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

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 Rareer1a, онлайн L_TE_X-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 =)

- ▶ номера страниц и логотип
- ▶ возможность писать сноски
- ▶ работа с заголовком
- ▶ прогрессбар

|| Посмотрим на код

Сделаем простой файл презентации

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

Создание команд

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

создаём команду

имя команды

число аргументов (может отсутствовать)

тело

|| Посмотрим на код

Создадим команду для быстрого набора “Computer Science Center”.

Пишем `\CSC`, получаем “Computer Science Center”

1. Запишем команду в .tex файл
2. Создадим пакет – файл стилей .sty
3. Перенесём команду в пакет

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

Внутри пакетов синтаксис меняется:

<code>\newcommand</code>	→	<code>\providecommand</code>
<code>\usepackage</code>	→	<code>\RequirePackage</code>
<code>\documentclass</code>	→	<code>\LoadClass</code>

Специальный синтаксис внутри пакетов

Внутри пакетов синтаксис меняется:

<code>\newcommand</code>	→	<code>\providecommand</code>
<code>\usepackage</code>	→	<code>\RequirePackage</code>
<code>\documentclass</code>	→	<code>\LoadClass</code>

→ Обратите внимание на нотацию двух последних
← команд

Специальный синтаксис внутри пакетов

Внутри пакетов синтаксис меняется:

<code>\newcommand</code>	→	<code>\providecommand</code>
<code>\usepackage</code>	→	<code>\RequirePackage</code>
<code>\documentclass</code>	→	<code>\LoadClass</code>

→ Обратите внимание на нотацию двух последних
← команд

|| Посмотрим на код

Изменим создание команду создания команды `\CSC`
на `\providecommand`,
Создадим новую команду, выделяющую текстовый
фрагмент по цвету **Вот так вот**

|| Посмотрим на код

Изменим создание команду создания команды `\CSC` на `\providecommand`,
Создадим новую команду, выделяющую текстовый фрагмент по цвету **Вот так вот**

1. Подключим пакет для работы с цветами
2. Определим нужные цвета
3. Создадим команду работы с цветом

|| Посмотрим на код

Изменим создание команду создания команды `\CSC` на `\providecommand`,
Создадим новую команду, выделяющую текстовый фрагмент по цвету **Вот так вот**

1. Подключим пакет для работы с цветами
2. Определим нужные цвета
3. Создадим команду работы с цветом

Вопрос: как сделать линейку сверху слайда? (Как прогрессбар)

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Простое создание команд

Длины: единицы измерения

Боксы и клей




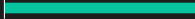
Моды и создание параграфов

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

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

Арифметика с длинами

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

|| Посмотрим на код

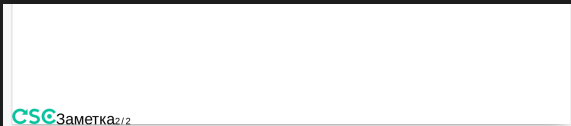
Пустим по верху линейку как в прогрессбаре

1. Модифицируем `headline`
2. Сделаем ведущий цвет зелёным
3. Используем линейку нужной длины командой `\rule`



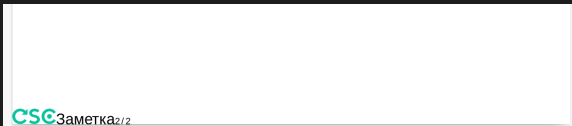
Посмотрим на код

...И по низу пустим лого, заметки и отображение страниц



Посмотрим на код

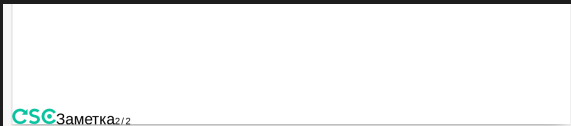
...И по низу пустим лого, заметки и отображение страниц



1. Модифицируем `footnline`
2. Перенесём файл логотипа в рабочую папку
3. добавим логотип с помощью `\includegraphics`
4. добавим номер слайда с `\insertframenumbers`
5. Создадим команду для добавления заметки

Посмотрим на код

...И по низу пустим лого, заметки и отображение страниц



1. Модифицируем `footnline`
2. Перенесём файл логотипа в рабочую папку
3. добавим логотип с помощью `\includegraphics`
4. добавим номер слайда с `\insertframenumbers`
5. Создадим команду для добавления заметки

Вопрос: Как сделать, чтобы длинные заметки не уводили номер страницы за экран?

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**

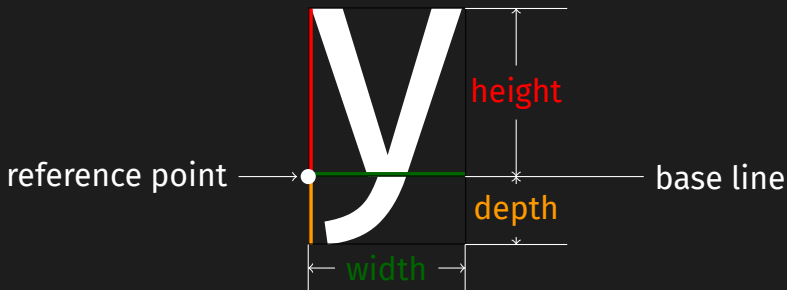
Параметры бокса: высота, глубина,
ширина

у

Параметры бокса: высота, глубина, ширина



Параметры бокса: высота, глубина, ширина



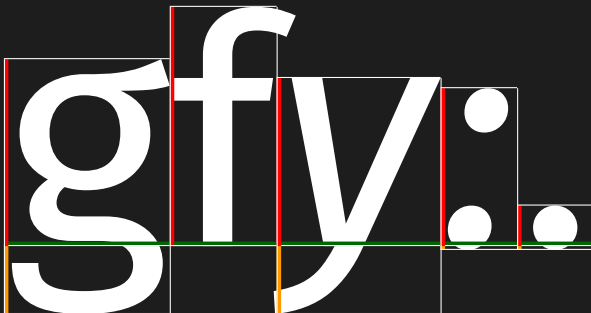
Как T_EX объединяет боксы



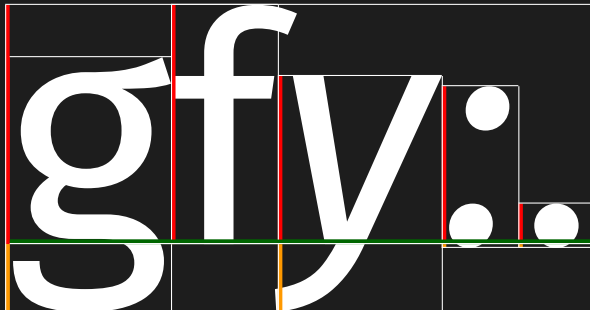
Как T_EX объединяет боксы



Как T_EX объединяет боксы



Как T_EX объединяет боксы



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

Использование

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

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

hello world!



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`

ΛT_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

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

Посмотрим на код

Теперь можем ограничить размер текста!

Заметка Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis
aute irure

CSC

2/2

Посмотрим на код

Теперь можем ограничить размер текста!

CSC

Заметка Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis
aute irure

2/2

- ▶ $\text{T}_{\text{E}}\text{X}$ -путь:
 1. Обернём все элементы в боксы
 2. Ограничим ширину заметки с помощью `\hspace=...`
 3. Поднимем заметку выше
- ▶ $\text{A}_{\text{T}}\text{E}_{\text{X}}$ -путь:
 1. Обернём все элементы в нужные боксы

Посмотрим на код

Теперь можем ограничить размер текста!

Заметка Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis
aute irure

CSC

2/2

- ▶ TeX-путь:
 1. Оборнём все элементы в боксы
 2. Ограничим ширину заметки с помощью `\hspace=...`
 3. Поднимем заметку выше
- ▶ L^AT_EX-путь:
 1. Оборнём все элементы в нужные боксы

Вопрос: как сделать, чтобы между элементами было пустое пространство?

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.

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 (L^AT_EX). `\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.

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 1fill\  
  relax fill vs fil \hskip0em  
  plus 999fil\relax}  
\hbox to 50mm{\hskip0em plus 1filll\  
  relax filll and filll \hskip0  
  em plus 1filll\relax}  
\hbox to 50mm{\hskip0em plus 1filll\  
  relax filll 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*.

|| Посмотрим на код

Добавим бесконечный клей между элементами

CSC

1/3

Посмотрим на код

Добавим бесконечный клей между элементами

CSC

1/3

1. Добавим `\hfill` между лого и заметкой, между заметкой и номером слайда
2. Добавим немного вертикального клея, чтобы отодвинуть текст от нижнего края
3. Добавим немного горизонтального клея перед лого, чтобы отодвинуть его от левого края

|| Посмотрим на код

И давайте изменим отображение заголовка

Заголовок

и подзаголовок

Посмотрим на код

И давайте изменим отображение заголовка

Заголовок

и подзаголовок

1. Модифицируем `frametitle`
2. Добавим заголовок `\insertframetitle`,
подзаголовок `\insertframesubtitle`
3. Проведём черту под подзаголовком, чтобы
отделить его зрительно `\rule`

Посмотрим на код

И давайте изменим отображение заголовка

Заголовок

и подзаголовок

1. Модифицируем `frametitle`
2. Добавим заголовок `\insertframetitle`,
подзаголовок `\insertframesubtitle`
3. Проведём черту под подзаголовком, чтобы
отделить его зрительно `\rule`

Проблема: плохо, что чёрточка под заголовком не прижимается к нему, когда нет подзаголовка

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>`ы игнорируются

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

|| Создание параграфов

... но посмотрите его дома :)

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. Теперь параметр `foobar` будет доступен как `\KVCSC@foobar`

`\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

|| Посмотрим на код

Реализуем возможность менять логотип через параметр пакета

|| Посмотрим на код

Реализуем возможность менять логотип через параметр пакета

1. Перенесём новый файл логотипа в рабочую папку
2. Подключим пакет
3. Создадим опцию `logo`
4. Передадим эту опцию `\KVCSC@logo` в создание команды `\logoname`

|| За эту часть мы узнали

- ▶ Как создавать стилевые файлы
- ▶ Как сделать футлайн и заголовок
- ▶ Как работать с боксами
- ▶ Как добавить клей
- ▶ Как создавать команды и передавать параметры в пакеты

Фактически мы умеем создать любую картинку

|| За эту часть мы узнали

- ▶ Как создавать стилевые файлы
- ▶ Как сделать футлайн и заголовок
- ▶ Как работать с боксами
- ▶ Как добавить клей
- ▶ Как создавать команды и передавать параметры в пакеты

Фактически мы умеем создать любую картинку, если она статична

|| Пока мы не знаем

- ▶ Как убрать номер слайда с титульника
- ▶ Как пододвинуть чёрточку под заголовком ближе к нему, когда подзаголовок нет
- ▶ Как делать динамические элементы типа прогрессбара
- ▶ Как можно манипулировать элементами

|| Пока мы не знаем

- ▶ Как убрать номер слайда с титульника
- ▶ Как пододвинуть чёточку под заголовком ближе к нему, когда подзаголовка нет
- ▶ Как делать динамические элементы типа прогрессбара
- ▶ Как можно манипулировать элементами

Т.е. для полноты картины нужно научиться работать с:

- ▶ условными операторами
- ▶ переменными
- ▶ циклами

|| Пока мы не знаем

- ▶ Как убрать номер слайда с титульника
- ▶ Как пододвинуть чёрточку под заголовком ближе к нему, когда подзаголовок нет
- ▶ Как делать динамические элементы типа прогрессбара
- ▶ Как можно манипулировать элементами

Т.е. для полноты картины нужно научиться работать с:

- ▶ условными операторами
- ▶ переменными
- ▶ циклами

научиться программировать на T_EX!

Итак, сегодня...

Слегка продвинутый \LaTeX : Типографика и создание команд

Очень продвинутый \LaTeX : программирование и работа с примитивами

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

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 #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!
 $\frac{\partial g}{\partial y}$

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

```
\def\dfdx #1 #2 {\frac{\partial #1}{\partial #2}}
```

команда задания макроса

имя макроса

первый аргумент

второй аргумент

что делать с аргументами

Pattern matching

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

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` выдаст `arg1: 'Hello ,,' arg2: 'World '`

|| Посмотрим на код

Мы часто можем ссылаться на ГитХаб. Давайте напишем команду, что сокращала бы такую ссылку

Код будет доступен на

 /Lavton/latexLectures

Посмотрим на код

Мы часто можем ссылаться на ГитХаб. Давайте напишем команду, что сокращала бы такую ссылку

Код будет доступен на



/Lavton/latexLectures

1. Перенесём иконку ГитХаба в рабочую папку
2. Создадим команду, которая преобразует `https://github.com/Lavton/latexLectures` в `/latexLectures`
3. Создадим обёртку над командой, которая бы добавляла иконку гитхаба

|| Пора добавить динамики в отображение!

Условные операторы

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
```

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

|| Посмотрим на код

Уберём номер слайда с титульника, подвинем линейку к заголовку (когда нет подзаголовка)

|| Посмотрим на код

Уберём номер слайда с титульника, подвинем линейку к заголовку (когда нет подзаголовка)
Вдруг у нас есть проблема: посчитать количество “сложных” слайдов. Как это сделать?

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=73`
- ▶ номер страницы при этом – `\thesection=100`
`\thepage=100`

Определение и манипулирование счётчиками

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



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`.
 - ▶ Эти команды меняют значение счётчика, к которому приложены
 - ▶ Просто арифметическую операцию можно сделать с помощью `\numexpr`: `\numexpr 4*3-\the\count0=-92`
- ▶ **Показать значение** `\the\<countname>` или `\number` или `\romannumeral`

|| Посмотрим на код

Добавим индикатор, что слайд “сложный” и выведем число таких сложных слайдов

|| Посмотрим на код

Добавим индикатор, что слайд “сложный” и выведем число таких сложных слайдов

Вопрос: как бы сделать, чтобы для слайдов 16:9 заметки внизу могли быть шире, чем для 4:3?

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`, работающая по аналогии с `\numexpr`.
- ▶ **Показать длину** `\the<lenname>`

|| Посмотрим на код

Вынесем ширину footnote в отдельную переменную.
И заодно длину линии по верху.

|| Посмотрим на код

Вынесем ширину footnote в отдельную переменную.
И заодно длину линии по верху.

А теперь определим ширину в зависимости от
ширины слайда.

Посмотрим на код

Вынесем ширину footnote в отдельную переменную.
И заодно длину линии по верху.

А теперь определим ширину в зависимости от ширины слайда.

1. Введём переменную со значениями 0, 1. Она отвечает за то, “широк” ли слайд
2. В зависимости от значения переменной присвоим длине заметки 9 или 12 cm

Посмотрим на код

Вынесем ширину footnote в отдельную переменную. И заодно длину линии по верху.

А теперь определим ширину в зависимости от ширины слайда.

1. Введём переменную со значениями 0, 1. Она отвечает за то, “широк” ли слайд
2. В зависимости от значения переменной присвоим длине заметки 9 или 12 cm

Проблема: у нас бывают очень большие заметки внизу. Было бы здорово, если заметка очень уж большая, расположить её по всему низу, убрав номер слайда и лого.

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`)

Посмотрим на код

Сделаем, чтобы широкая заметка могла занять весь “подвал”. И тогда она займёт не так много места по вертикали

Заметка настолько длинная, что ради неё стоит убрать номер слайда и лого Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation

Посмотрим на код

Сделаем, чтобы широкая заметка могла занять весь “подвал”. И тогда она займёт не так много места по вертикали

Заметка настолько длинная, что ради неё стоит убрать номер слайда и лого Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation

1. Зададим новый бокс, в котором будет храниться заметка
2. Создадим функцию, которая бы проверила – выше ли заметка некого порога
3. Создадим именованный if, значение которого бы показывало – нужно нам заметку расширять на весь слайд или нет
4. В footline проверим значение if-а и или расширим заметку или нет

|| Мы почти у финиша!

Итак, мы умеем делать уже почти всё. Одно лишь загадка: как сделать прогрессбар?

|| Мы почти у финиша!

Итак, мы умеем делать уже почти всё. Одно лишь загадка: как сделать прогрессбар?

... для него нужны циклы, рекурсия и даже работа с файловой системой!

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 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?



Циклы в L^AT_EX

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

A? A? A? A?

```
\usepackage{forloop}
```

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

- ▶ `\newrite` 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}`
- ▶ `\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

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\{ooo} \uppercase{\a, my oborona}` вернёт “ooo, MY OBORONA”
 - ▶ `\def\{ooo} \uppercase\expandafter{\a, my oborona}` вернёт “OOO, 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

Манипулирование именами команд



- ▶ Написать `\csname textit\endcsname` всё равно, что написать команду `\textit`:
`\csname textit\endcsname{my text}` даст *my text*
- ▶ `\string\textit` вернёт просто `\textit`

Манипулирование именами команд



- ▶ Написать `\csname textit\endcsname` всё равно, что написать команду `\textit`:
`\csname textit\endcsname{my text}` даст *my text*
- ▶ `\string\textit` вернёт просто `\textit`

Пример использования:

- ▶ Зададим команду `\def\myfontchange#1{\csname text#1\endcsname}`
- ▶ `\myfontchange{it}{hello}` даст *hello*
- ▶ `\myfontchange{bf}{world}` даст **world**



Catcodes

что отвечает за комментарий, а что – символ команды?

```
\catcode'\%=12
```

Меняем категорию символа

Символ – %. Раньше отвечал за комментарий
новый код символа – 12. Обычный текст

14 – код комментария, 1 – начало группы {}, и т.д...

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

[{{aasd}}]
]
```

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



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

`\message` и `\typeout`

Пишет сообщение в лог:

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

|| Посмотрим на код

На этом теория закончилась. Пора применить все знания и создать прогрессбар!

Что сегодня узнали

1. Примитивы в \LaTeX

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

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