

Отчет по лабораторной работе №1 : L^AT_EX, Git, GPG

Дедков Сергей

2015

Содержание

1	Система верстки T_EX и расширения L^AT_EX	3
1.1	Цель работы	3
1.2	Ход работы	3
1.2.1	Создание минимального файла .tex в простом текстовом редакторе – преамбула, тело документа . . .	3
1.2.2	Компиляция в командной строке – latex, xdvi, pdf _l atex	3
1.2.3	Оболочка TexMaker, Быстрый старт, Быстрая сборка	4
1.2.4	Создание титульного листа, нескольких разделов, списка, несложной формулы	5
1.2.5	Понятие классов документов, подключение пакетов	7
1.2.6	Верстка более сложных формул	7
1.3	Выводы	8
2	Система контроля версий Git	8
2.1	Цель работы	8
2.2	Ход работы	8
2.2.1	Изучить справку для основных команд	8
2.2.2	Получить содержимое репозитория	8
2.2.3	Добавление новой папки и первого файла под контроль версий	9
2.2.4	Зафиксировать изменения в локальном репозитории	9
2.2.5	Внести изменения в файл и просмотреть различия .	9
2.2.6	Отменить локальные изменения	9
2.2.7	Зафиксировать изменения в локальном репозитории, зафиксировать изменения в центральном репозитории	9
2.2.8	Получить изменения из центрального репозитория .	10
2.2.9	Поэкспериментировать с ветками	10
2.3	Выводы	10

3	Программа для шифрования и подписи GPG, пакет Gpg4win	10
3.1	Цель работы	10
3.2	Ход работы	10
3.2.1	Изучить документацию, запустить графическую оболочку Kleopatra	10
3.2.2	Создать ключевую пару OpenPGP (File->New Certificate)	10
3.2.3	Экспортировать сертификат (File->Export Certificate)	11
3.2.4	Поставить ЭЦП на файл (File->Sign/Encrypt Files)	11
3.2.5	Получить чужой сертификат из репозитория, файл с данными и файл с сигнатурой(подписью)	11
3.2.6	Импортировать сертификат, подписать его	11
3.2.7	Проверить подпись	11
3.2.8	Взять сертификат кого-либо из коллег, зашифровать и подписать для него какой-либо текст, предоставить свой сертификат, убедиться, что ему удалось получить открытый текст, проверить подпись.	12
3.2.9	Предыдущий пункт наоборот	12
3.2.10	Изучить GNU Privacy handbook, протестировать в использовании gpg через интерфейс командной строки, без использования графических оболочек.	12
3.3	Выводы	13

1 Система верстки T_EX и расширения L^AT_EX

1.1 Цель работы

Изучение принципов верстки T_EX, создание первого отчета

1.2 Ход работы

1.2.1 Создание минимального файла .tex в простом текстовом редакторе – преамбула, тело документа

Структура самого простого документа L^AT_EX выглядит следующим образом:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[russian]{babel}
\begin{document}
Hello!!
\end{document}
```

Создадим файл test.tex. И введем в него данный код.

После компиляции подобный документ выведет просто слово Hello!!

Рассмотрим подробнее структуру документа:

Документ делится на две части: *преамбула* и *тело* документа.

Преамбула - часть с которой начнется документ(с начала до `\begin{document}`), в которой вы задаете класс документа, его классовые опции, параметры страницы, подключаете пакеты, которые собираетесь использовать при верстке. Так, каждый входной файл должен начинаться с команды

```
\documentclass{...}
```

Она указывает, документ какого типа вы собираетесь писать. После этого можно включать команды, влияющие на стиль документа в целом, или загружать пакеты, с новыми возможностями. Для загрузки такого пакета используется команда

```
\usepackage{...}
```

Далее следует *тело* документа - начинается с `\begin{document}` и заканчивается `\end{document}`. В тело документа пишется текст документа.

1.2.2 Компиляция в командной строке – latex, xdv, pdflatex

Скомпилируем файл test.tex в командной строке.

Для этого воспользуемся командой latex:

```
latex test.tex
```

После компиляции в той же папке будет создан файл test.dvi.

DVI (от англ. DeVice Independent — аппаратно независимый) — формат выходных файлов издательской системы. Далее .dvi файлы преобразуются в другие читабельные форматы.

Преобразовать его можно, например, в pdf, используя команду dvipdfm.

Чтобы создать pdf файл сразу из tex файла можно воспользоваться командой pdflatex.

1.2.3 Оболочка TexMaker, Быстрый старт, Быстрая сборка

Texmaker это кросс-платформенный open source \LaTeX редактор с интегрированной программой для просмотра PDF-файлов, написанный на Qt. Оболочка TexMaker удобна в использовании включает в себя множество настроек.

Texmaker с помощью функции Быстрый старт позволяет сформировать преамбулу документа, используя GUI. Ниже на рисунке представлены образец формы Быстрый старт.

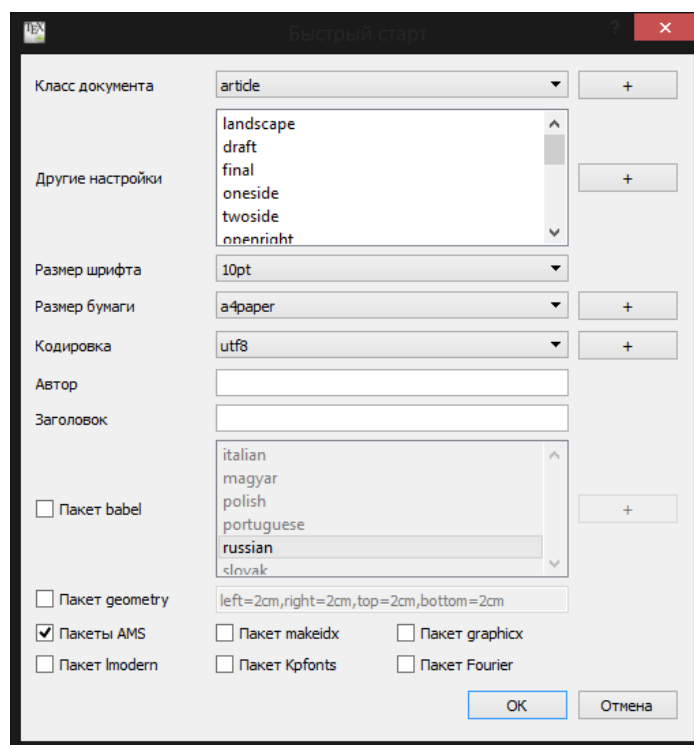


Рис. 1: Окно Быстрый старт

Самый простой способ скомпилировать документ — это использовать команду "Быстрая сборка". Задать последовательность команд используемых командой "Быстрый старт" можно в диалоге "Настроить

Texmaker". Для запуска команды из панели инструментов сначала выберем ее, а затем нажмем кнопку "Run".

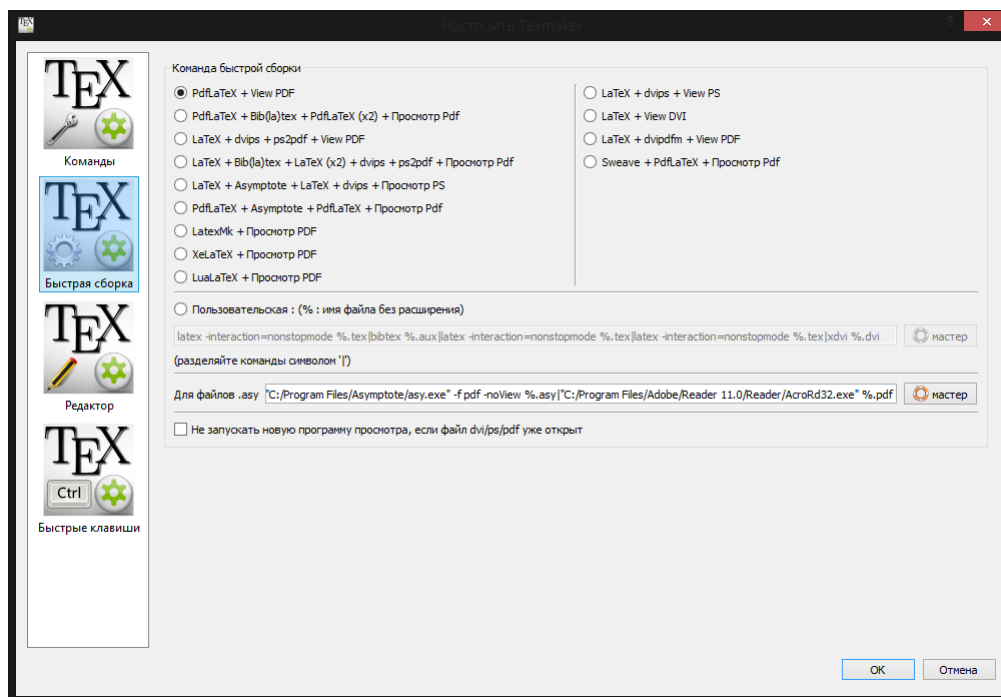


Рис. 2: Окно Быстрый старт

1.2.4 Создание титульного листа, нескольких разделов, списка, несложной формулы

Для создания титульного листа в преамбулу документа следует добавить следующие команду `\title{...}`. Так же опционально можно указать автора и дату: `\author{...}`, `\date{...}`. Пример:

```
\author{Jane Smith}
\title{My article}
\date{2015}
```

Чтобы титульный лист отобразился нужно уже в теле документа указать команду: `\maketitle{...}`. Так же можно указать команду разрыва страницы, чтобы титульный лист отобразился на отдельной странице `\newpage{...}`.

Пример документа с титульным листом:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
```

```
\usepackage[russian]{babel}
```

```
\author{Jane Smith}
```

```
\title{My article}
```

```
\date{2015}
```

```
\begin{document}
```

```
\maketitle
```

```
\newpage
```

```
Hello!!
```

```
\end{document}
```

Для создания разделов используются команды `\section{...}`, `\subsection{...}`, `\subsubsection{...}` (секция, подсекция, под-подсекция соответственно).

Следует отметить, что их следует использовать в правильном порядке.

Для выделения параграфов используются команды: `\paragraph{...}` и `\subparagraph{...}`

При разбиении документа на разделы есть возможность автоматического создания оглавления с помощью команды `\maketitle`.

Окружение `itemize` подходит для создания простых списков, окружение `enumerate` - для нумерованных, а окружение `description` - для описаний.

Пример:

```
\flushleft
```

```
\begin{enumerate}
```

```
\item Вы можете как угодно смешивать окружения списков:
```

```
\begin{itemize}
```

```
\item Но это может смотреться глупо.
```

```
\item[-] С минусом.
```

```
\end{itemize}
```

```
\item Поэтому помните:
```

```
\begin{description}
```

```
\item[Глупые] вещи не станут умнее от помещения в список.
```

```
\item[Умные] вещи, однако, вполне можно представить списком.
```

```
\end{description}
```

```
\end{enumerate}
```

Для отображения математических формул применяются следующие конструкции:

- `$. . . $` - для формул внутри текста. Результат: $a^2 + b^2 = c^2$
- `\[. . .\]` - для формул текст которых следует начать по центру в новой строке. Результат:

$$a^2 + b^2 = c^2$$

1.2.5 Понятие классов документов, подключение пакетов

Как было отмечено выше в документе существует служебная часть, называемая преамбулой. В этой части указывается помимо прочего класс документа и подключаемые пакеты. Класс определяет тип документа, пакеты расширяют возможности L^AT_EX.

Классы документов:

- **article** - статьи в журналах, краткие отчеты, презентации, приглашения...
- **report** - для больших длинных отчетов, содержащих несколько глав, небольших книг, диссертаций...
- **book** - для книг.
- **slides** - для слайдов. Использует большие буквы без засечек. Вместо этого можно использовать FoilT_EX.

Пакет активизируются командой `\usepackage[опции]пакет`. *Пакет* - имя пакета. *Опции* - список ключевых слов, включающий специальные свойства пакета. Некоторые пакеты включены в основную поставку, другие предоставляются отдельно.

1.2.6 Верстка более сложных формул

Существует возможность задавать различные стили оформления форм: Мы уже попробовали создать простую включенную и выключенную формулы. Теперь создадим что-то более сложное. Для ясного задания стиля оформления формул используются четыре команды:

- Выключенная формула - `\displaystyle: \iint_D g(x, y) \, dx \, dy`
- Текстовая формула - `\textstyle: \iint_D g(x, y) \, dx \, dy`
- Индексная формула - `\scriptstyle: \iint_D g(x, y) \, dx \, dy`
- Подиндексная формула - `\scriptscriptstyle: \iint_D g(x, y) \, dx \, dy`

Для написания тех или иных формул существуют соответствующие команды. Их достаточно много, поэтому не имеет смысла все перечислять. Достаточно воспользоваться справочным пособием при написании формул и найти там необходимые команды. Пример нескольких сложных формул ниже:

- Предел:

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$$

- Матрица:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots \\ x_{21} & x_{22} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

- Оператор суммы:

$$\sum_{\substack{0 < i < n \\ 1 < j < m}} P(i, j)$$

1.3 Выводы

Л^AT_EX - очень удобный инструмент. В освоении может сначала показаться сложным, но после кратковременного использования начинает нравиться. Плюс того, что верстка задается в коде является возможность одновременного редактирования несколькими людьми один документ, после чего совмещать изменения используя системы контроля версий.

2 Система контроля версий Git

2.1 Цель работы

Изучить систему контроля версий Git, освоить основные приемы работы с ней.

2.2 Ход работы

2.2.1 Изучить справку для основных команд

Основные команды которые следует знать: `init`, `clone`, `status`, `commit`, `pull`, `push`, `switch`, `checkout`, `branch`. Ресурс для ознакомления <https://git-scm.com/> - здесь можно найти примеры применения этих команд, а так же пошагово научиться работать с СКВ git. Ниже будут рассмотрены случаи применения этих команд.

2.2.2 Получить содержимое репозитория

Для получения содержимого репозитория используется команда `clone`.

Синтаксис: `git clone <repository>`

`<repository>` - URL к репозиторию, он может быть как локальный, так и через сеть интернет.

Ниже приводится пример получения содержимого репозитория github.

2.2.3 Добавление новой папки и первого файла под контроль версий

Для начала отметим, что пустые папки не добавляются под контроль версий, контролируются только файлы.

После создания папки и внутри нее файла система контроля версий автоматически начинает контролировать этот файл, если только в файле .gitignore не прописано иного. Как формируется файл .gitignore можно найти на просторах сети интернет, воспользовавшись, например, поисковой системой google.

Теперь, если после добавления не совершалось никаких действий, в частности, если пользователь не вызывал команду commit.

Посмотреть добавленные файлы можно воспользовавшись командой status.

2.2.4 Зафиксировать изменения в локальном репозитории

Чтобы зафиксировать изменения в локальном репозитории (то, что у пользователя на компьютере, пока что не внося изменений в удаленный репозиторий), следует воспользоваться командами:

- git add - для добавления файла в текущую ревизию. Это нужно для того, чтобы, например, не вносить всех изменений за одну ревизию, а разделить на этапы. Сначала добавить одни файлы, затем другие. В рамках этого примера добавляются все файлы.
- git commit - фиксация изменений в ревизии, в локальном репозитории.

2.2.5 Внести изменения в файл и просмотреть различия

Если теперь изменить файл и воспользоваться после этого командой git status, измененный файл будет помечен, как modified.

Для просмотра конкретных изменений существует команда: git diff

2.2.6 Отменить локальные изменения

Может случиться, так, что сделанные изменения не устраивают пользователя и ему хотелось бы вернуть файл или несколько файлов к исходному состоянию.

Для этого существует команда: git checkout -- <filename>

2.2.7 Зафиксировать изменения в локальном репозитории, зафиксировать изменения в центральном репозитории

Для фиксации изменений необходимо вызвать команду. git commit -a -m "commit message".

В центральном репозитории изменения фиксируются командой `git push`.

2.2.8 Получить изменения из центрального репозитория

Получение изменений из центрального репозитория - `git pull`.

2.2.9 Поэкспериментировать с ветками

Создание ветки - `git checkout -b`.

Переключение веток - `git checkout`.

Совмещение изменений в ветках - `git merge`.

2.3 Выводы

Система контроля версий `git` позволяет осуществлять командную разработку. Удобная в использовании и легкая в освоении. Так же можно использовать помимо командной строки - программы, которые предоставляют пользовательский интерфейс.

3 Программа для шифрования и подписи GPG, пакет Gpg4win

3.1 Цель работы

Научиться создавать сертификаты, шифровать файлы и ставить ЭЦП.

3.2 Ход работы

3.2.1 Изучить документацию, запустить графическую оболочку Kleopatra

В ходе выполнения лабораторной работы была установлена и освоена графическая оболочка Kleopatra.

3.2.2 Создать ключевую пару OpenPGP (File->New Certificate)

Была создана ключевая пара OpenPGP (PGP/MIME). Для этого в главном меню выбираем File->New Certificate. Далее из предложенных вариантов указываем, что необходимо создать ключевую пару OpenPGP. После этого вводим имя, электронную почту и passphrase. Далее ждем пока будет создана пара ключей Open PGP и нажимаем Finish.

3.2.3 Экспортировать сертификат (File->Export Certificate)

Экспортируем сертификат выбрав в главном меню File->Export Certificate и указав папку, куда необходимо произвести экспорт. В этой папке будет создан файл с расширением .asc, который содержит открытый ключ. Теперь этот файл можно передать партнеру, для возможности передавать сообщения и файлы в зашифрованном виде.

3.2.4 Поставить ЭЦП на файл (File->Sing/Encrypt Files)

Что бы подписать файл выберем в главном меню пункт File->Sing /Encrypt Files. Теперь укажем файл, который необходимо подписать и выберем пункт Sing. Указываем сертификат, которым производится подпись и файла. Теперь подтверждаем выбор и нажимаем Sing. Вводим passphrase в диалоговом окне PIN-кода. После того, как подпись будет создана, появится файл подписи с расширением .sig.

3.2.5 Получить чужой сертификат из репозитория, файл с данными и файл с сигнатурой(подписью)

Из репозитория возьмем файл с данными - myfirst.pdf, файл подписи - myfirst.pdf.sig и файл сертификата - karina.asc.

3.2.6 Импортировать сертификат, подписать его

Для импортирования сертификата в главном меню выбираем File->Import Certificates и выберем в качестве сертификата файл karina.asc. После чего сертификат будет добавлен в список импортированных сертификатов.

3.2.7 Проверить подпись

Для проверки подлинности и целостности файла myfirst.pdf нажмем на нем правой кнопкой, выберем пункт контекстного меню: Другие параметры GpgEX->Проверить. Для проверки целостности и подлинности необходимо, чтобы файл подписи лежал в одной папке с исходным файлом. Это позволит автоматически найти соответствие между ними. Подтвердим операцию нажатием Decrypt/Check. В случае успеха, будет выдано соответствующее сообщение. Если же в файле было произведено малейшее изменение, после совершения подписи, то подтверждение целостности пройдет неудачно.

3.2.8 Взять сертификат кого-либо из коллег, зашифровать и подписать для него какой-либо текст, предоставить свой сертификат, убедиться, что ему удалось получить открытый текст, проверить подпись.

Эксперимент был проделан мной самим используя на компьютере и виртуальной машине. В ходе которого удалось зашифровать файл на основной машине и расшифровать на виртуальной. Файлы .asc и from_alice_to_bob.txt.gpg прилагаются. Содержимое файла Hello, bob)

3.2.9 Предыдущий пункт наоборот

Мной был взят файл с открытым ключом и импортирован в систему Kleopatra. Далее получив файл from_bob_to_alice.txt.gpg, расшифровал и проверил подпись. Эксперимент прошел успешно. Содержимое файла: Hello, Alice).

3.2.10 Изучить GNU Privacy handbook, протестировать в использовании gpg через интерфейс командной строки, без использования графических оболочек.

Были изучены следующие команды:

- `-gen-key` – создание новой пары ключей. Команда `-edit-key` позволяет в последствии изменить параметры ключа.
- `-sign` – создает подпись для указанных файлов, используя ключ (если не введен его идентификатор, использует ключ по-умолчанию). `-detach-sign` создаст подпись в отдельном файле, иначе создается совмещенная подпись.
- `-encrypt` – указывает на то, что данные надо зашифровать. Если применяется совместно с `-sign` то данные одновременно подписываются.
- `-symmetric` – можно применять когда просто надо зашифровать файл (используется симметричный алгоритм, но можно выбрать алгоритм командой `-cipher-algo`).
- `-decrypt` – расшифровывает указанные файлы и сохраняет результат в файл (если указан) или выводит в консоль. Если файлы содержат подпись, она также проверяется.
- `-verify` – проверяет подписи для указанных файлов, не выводя содержимого файла.
- `-list-keys` – выводит список всех открытых ключей. Команда `-list-secret-keys` показывает ваши секретные ключи.

- `–delete-key` – удаляет открытый ключ из списка. `–delete-secret-key` удаляет секретный ключ (например, он скомпрометирован или окончился срок действия)
- `–export` (`–import`) – экспорт/импорт ваших ключей, например, для резервирования или переноса на другой компьютер. По умолчанию ключи в бинарном виде, для получения их в ASCII-виде укажите параметр `–armor`.

3.3 Выводы

В ходе выполнения третьего пункта лабораторной работы была освоена программа Kleopatra, входящая в пакет Gpg4win и используемая для шифрования и подписи GPG. Была создана пара ключей OpenPGP, в которой закрытый ключ защищен при помощи passphrase, а открытый ключ распространяется как сертификат(файл с расширением .asc). Произведен импорт нескольких сертификатов, что позволило зашифровать файлы, которые могут быть расшифрованы только владельцами этих сертификатов. Также была произведена подпись документов и соответственно проверка целостности и подлинности этих и других документов. Для проверки целостности и подлинности файла необходим сам файл, а также файл подписи.