Отчет по лабораторной работе №1 : LaTeX, Git, GPG

Бусаров Владислав

2015

Содержание

1 Система верстки TEX и расширения LATEX

1.1 Цель работы

Изучение принципов верстки TEX, 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 LATEX 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 компипляции подобный документ выведет просто слово Hello!!

Рассмотрим подробнее структуру документа:

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

Преамбула - часть с которой начинется 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 пишеться текст документа.

1.2.2 Компиляция в командной строке – latex, xdvi, 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 комманду dvipdfm.

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

1.2.3 Оболочка TexMaker, Быстрый старт, Быстрая сборка

Texmaker это кросс-платформенный open source LaTeX редатктор с интегррированной программой для просмотра PDF-файлов, написанный на Qt. Оболочка TexMaker удобна в использованиии включает в себя множество настроек.

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

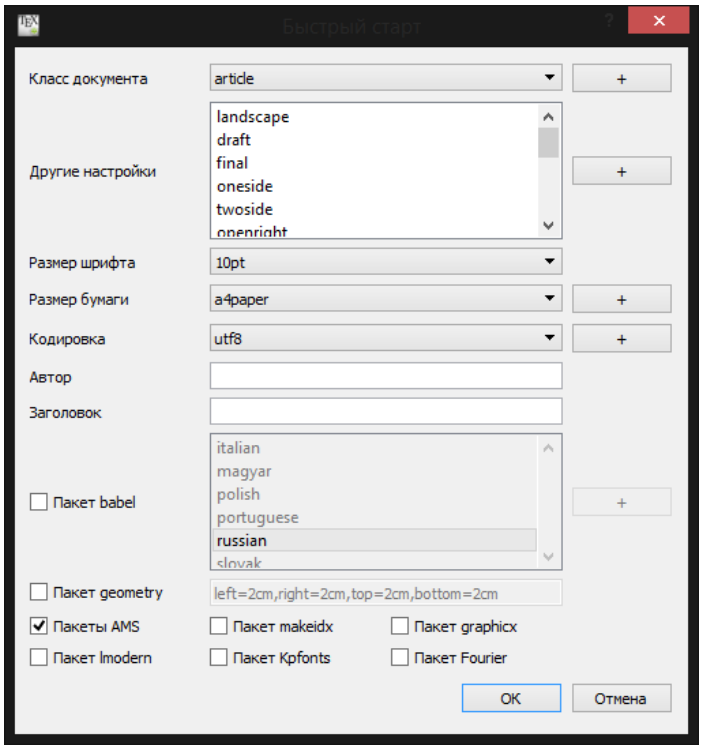

Рис. 1: Окно Быстрый старт

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".

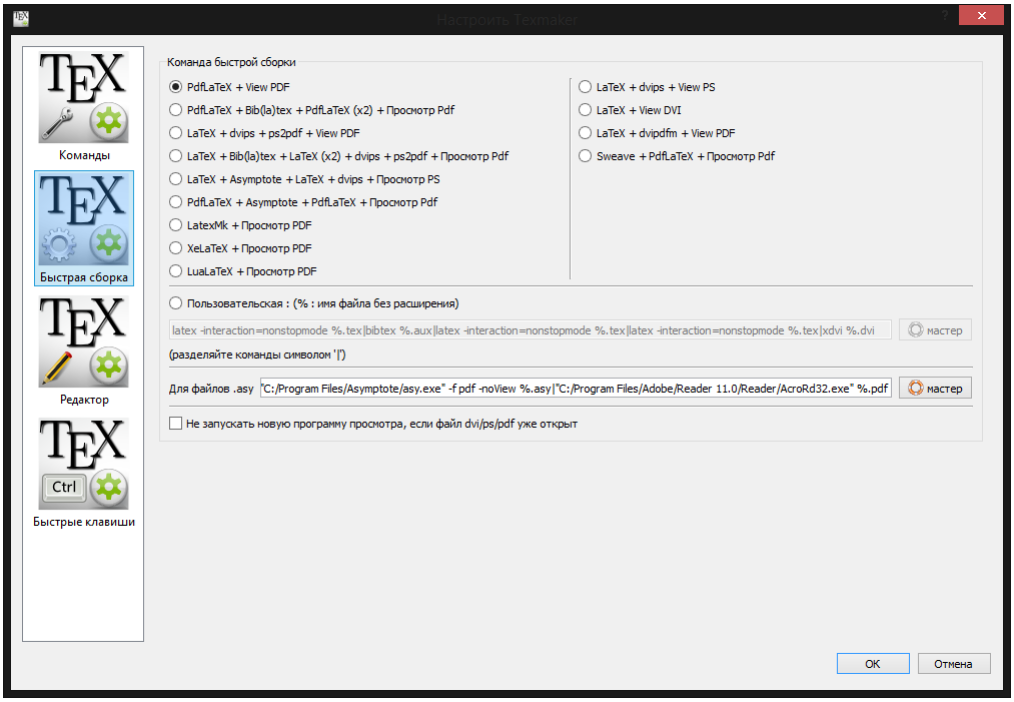

Рис. 2: Окно Быстрый старт

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

Для создания титульного листа в преамбулу документа следует добавть 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 разрава страницы, чтобы титульный лист отбразился 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 соответсвенно). 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 нумерованых, а окружение `descrioption` - 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 тектса. 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 LaTeX.

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 FoilTeX.

Пакет активизируются командой `\usepackage`[*опции*]*пакет*. *Пакет* - имя пакета. *Опции* - список ключевых слов, включающий специальные свойства пакета. Некоторые пакеты включены в основню 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 - `$\dispalystyle`: $\displaystyle\iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$
- Текстовая формула - `$\textstyle`: $\textstyle\iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$
- Индексная формула - `$\scriptstyle`: $\scriptstyle\iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$
- Подиндексная формула - `$\sciptscriptstyle`: $\scriptscriptstyle\iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$

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

- Предел:

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

- 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

LaTeX - очень удобный инструмент. В освоении может сначала показаться сложным, но после кратковременного использования начинает нравиться. Плюсом 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 комманд

Основные команды которые следует знать: init, clone, status, commit, pull, push, switch, checkout, branch. Ресурсс 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 привеодиться 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 формруется 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(тот 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 резделить на этапы. Сначала добавить одни файлы, затем другие. В рамках этого примера добавляются все файлы.
- git cimmit - фиксация изменений в ревизии, в локальном репозитории.

2.2.5 Внести изменения в файл и просмотреть различия

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

Для просмотра конкретных изменений существет 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 контролья 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)

Эскортируем 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 совершение 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, убедится, 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 ASCI-виде укажите параметр –armor.

Рассмотрим подробней механизм работы с gpg при использовании командной строки: Для начала сгенерируем ключ:

gpg –gen-key

gpg предложит выбрать тип ключа - возможные варианты:

```
Выберите тип ключа:
   (1) RSA и RSA (по умолчанию)
   (2) DSA и Elgamal
   (3) DSA (только для подписи)
   (4) RSA (только для подписи)
```

Выберем вариант по умолчанию - RSA и RSA.

Далее выбираем размер ключа - 2048.

```
Какой размер ключа Вам необходим? (2048)
Запрошенный размер ключа - 2048 бит
```

Затем срок действия - после того как пройдет этот период времени ключ станет недействительным. Выберем 0 - без ограничений.

```
Срок действия ключа? (0)
Срок действия ключа не ограничен
Все верно? (y/N) y
```

После этого программа попросит ввести личные данные и фразу-пароль. Введем эти данные:

```
Ваше настоящее имя: John_
Адрес электронной почты: john@yandex.ru
Комментарий: comment
Вы выбрали следующий ID пользователя:
    "John_ (comment) <john@yandex.ru>"
```

И наконец сгенерирует ключ:

```
gpg: проверка таблицы доверия
gpg: 3 ограниченных необходимо, 1 выполненных необходимо, модель доверия PGP
gpg: глубина: 0  корректных:   7  подписанных:   0  доверия: 0-, 0q, 0n, 0m, 0
 7u
pub   2048R/614E98E4 2015-05-25
      Отпечаток ключа = E7F4 E52E F0EB BF3F 6088  70C2 C9B0 CF07 614E 98E4
uid     [абсолютное] John_ (comment) <john@yandex.ru>
sub   2048R/DFC70E94 2015-05-25
```

Если теперь выполнить команду:

```
gpg --list-keys
```

то увидим в списке вновь созданный ключ:

```
pub   2048R/614E98E4 2015-05-25
uid     [абсолютное] John_ (comment) <john@yandex.ru>
sub   2048R/DFC70E94 2015-05-25
```

далее сделаем две вещи:

- зашифруем информацию, для того чтобы потом расшифровать
- подпишем файл для того, чтобы потом проверить менялся ли он(для этого далее проведем верификацию)

Шифрование. Для того, чтобы можно было в последующем на другой машине расшифровать/верифицировать наши файлы необходимо экспортировать ключ. Воспользуемся для этого ключем –export. Пример команды:

```
gpg --export -a John_
```

Увидим следующее:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQENBFVjNFgBCADLmmQ2AAjtdqRbZKvwbjSEhme9jFX8DXyJajDLh3gMCm8+vnjW
7zyXIuPQn7yD6hA7t/1u2x/ltFv4NOHUwFMgzYO+P/QcWin//9V3QKEZJRNqe6a8
X49W5+iJyXLULrFb8UebGJ2QeR+cyJ8HtQx2E3qsOz5OoBlXRjxjovUWvXNF9PzU
Ddw5oPKOnC2w9Lbg76jz+1AENo7aFNuWsaZOxkE291X5+GkeuyOcE/j9g329v+6K
OUZ8Oi4r3JumJ4OQLEnbjPUtVYKzuQxa8dOlNOngvOFKHL6NixOc7eOihnHoN/Zx
zUjnMKH1TuxLX23D1dGvTz4lkYliLMoEzKTDABEBAAG0IEpvaG5fIChjb21tZW50
KSA8am9obkB5YW5kZXgucnU+iQE5BBMBCAAjBQJVYzRYAhsDBwsJCAcDAgEGFQgC
CQoLBBYCAwECHgECF4AACgkQybDPB2FOmOS2mAgArjCYrh22t2JaF+Tvj7jJwonb
```

```
4hHMHX5joRQmWBmWLbhgj2V+KkcyR+14ZoW4zZ6itlOvK4bb3xbDjls36gxWCz93
isiJa4xJRweJnSSguxNo53HSPyBo8iq6xS9IjEA6zybJPhodD5934OXao857IuJo
NexvNB6KyowujQtbvLtfgCWcT/MNJoWP5YLB8A6olKkUdPIwVrvQxXVekg9QO+jB
mRO++RM1l+ZapQHzJID8nEBVaDaMLMovpr35CK2xSXFDMEhdBjOfievgepMx8p/8
g+uk6HFgvs8u6lBxhSRwWeQPkVsOSOuIvL7//1jvZWy7uWLszWtoco8IDCLDcbkB
DQRVYzRYAQgAvcPb77ah4DlTS8tcHjq8zTf6zkwm3hjYVLgPbNJUdSmFlSsP9KKG
gUSkjEp6uyN2Kg0DSu4g9c9SkdwO8CNL92KI4Ht+g1GLz+NhJT3Z2wyaiVCdtOn8
2JcVVlNplrZvq4BCBAX3/1RphIu+U37QxZ46UGU/iotOVCZpMnRB5VTzN9OMjCIU
/tNJEZKXqhaErY9M1Y8YLsVExvUnkJM9uVLt1LMssUyVYmMfUG8M9hQZU31AisFG
WIZPqZjKu1QOn78P4QzjL1XSO7RhBPboshc2ONu33wgEJExqHUUszaMbocE4Od7i
3I6bEFzoS8T8YEQ/u8Lma2aIgokVUTRuJQARAQABiQEfBBgBCAAJBQJVYzRYAhsM
AAoJEMmwzwdhTpjkOwQH/OuEsZReyTbdifoniiKUyOh9ZqOQWBQsa9gzcIMOoyr+
Ku+QBhFwtnroQ1MNjCOn/ysoAb1pWmgPPf2N5hYit54xS3OxGOfcBfRnUh1S/YxE
D5SvIM3GjoEe2k4lKQkNfoA7lPFMdgSkloLHKZK2JRcraq7uGUoY6ufakhPXX8ql
UUyLclo4tJyx43Drbnpql5aJ3HIQxhUD6xUBA4ONluoWpkfY8ZfwXCEU1WPxsFnh
wNbqLDHx2hG2x8152lxHVKcMj22RdhK5KBrVA5lfujxyGzFLjc1N4QJztqCwqNbJ
+Kqg5vA4mzS5+vnr9Fy4sQ9QmqTmhhXelsjuEqVhg9k=
=YDll
-----END PGP PUBLIC KEY BLOCK-----
```

Сделаем вывод в файл, чтобы потом отправить его на удаленную машину:

```
gpg --export -a John_ >John.asc
```

В текущей папке появиться файл John.asc. Отправим его на удаленную машину после чего импортируем. Команда импорта и результат:

```
root@ubuntu:~/Загрузки$ gpg --import John.asc
gpg: ключ 614E98E4: открытый ключ "John_ (comment) <john@yandex.ru>" импортирован
gpg: Всего обработано: 1
gpg:               импортировано: 1  (RSA: 1)
```

Теперь если посмотреть список ключей увидим в том числе и наш:

```
root@ubuntu:~/Загрузки$ gpg --list-keys
/root/.gnupg/pubring.gpg
------------------------
pub   2048R/48E59386 2015-05-13
uid                  Bob__ <bob@yandex.ru>

pub   2048R/2AD7FC08 2015-05-11
uid                  Alice <alice@gmail.com>
```

```
pub   1024D/7C77ACF9 2015-05-25
uid                  Dima_ (comment) <dima@yandex.ru>
sub   1024g/2B1C5A2D 2015-05-25

pub   2048R/614E98E4 2015-05-25
uid                  John_ (comment) <john@yandex.ru>
sub   2048R/DFC70E94 2015-05-25
```

Теперь перейдем к шифрованию: зашифруем файл на удаленной машине, после чего скинем его на основную и расшифруем. Файл - message_to_john. Содержание:

```
Hello, John!
```

Выполним команду:

```
root@ubuntu:~/Загрузки$ gpg -ea -r John_ message_to_john
```

В папке появиться файл: message_to_john.asc.
Содержание:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1

hQEMAy7QOnvfxw6UAQf+K/OhgvOl8SfK7z8+B3KGmbZWhfNOinUKsstCH6YKYtu2
nrkyhFs8wjBKPcW6ozSVPF2o+gPaFWe8eugrMuB5YmoVEWWdCadOJnsgtAGGVlsR
6vcuEt13mOoM6J+Bp7DSB+aeEvUtXzHVa3C16+SDE7ZqVuevtzWXAa29IQygGMMn
LrLPTlcCAVeZlNoXpvOIDyHG3qzsP52Xrr9y2IY/lfHTYuLvoAyJUpa+KMEm8zdd
gG+JiuGZTZaX8w/LtyBSLx67FAZwS/B38bp3rJ6XRbDnH7RNugRadh6AH9YGYAT+
ZQf/1CkBk61chfuL4HTkHzOSWFUxuGKw+c6UYXdNDNJWASMGI6Xho2IXfE9QPZ3E
ZPSmXiVsq58cWO9W4mX4h79l9SLeTmVBaVsVzh7kiN6epM4iuM+7fCW8K8x7uzmL
oVtnxVrMIlSysLBy5MpO7rAhdIzI3HI=
=9k5I
-----END PGP MESSAGE-----
```

Скинем его на основную машину, чтобы расшифровать закрытым ключем. Введем команду:

```
gpg -d message_to_john.asc >message_to_john.txt
```

Прогамма ответит и потребует ввести фразу-пароль:

```
Необходим пароль для доступа к закрытому ключу пользователя: "John_ (comment)
ohn@yandex.ru>"
2048-битный ключ RSA, ID DFC70E94, создан 2015-05-25 (ID главного ключа 614E98
```

```
)

gpg: зашифровано 2048-битным ключом RSA, с ID DFC70E94, созданным 2015-05-25
      "John_ (comment) <john@yandex.ru>"
```

Тем самым получим исходный файл. С содеражнием:

```
Hello, John!
```

Цифровая подпись. Далее рассмотрим другой вариант использования gpg, когда нужно не шифровать файл а ставить электронную подпись. На удаленной машине введем команду:

```
gpg --clearsign -a message\_to\_john
```

Получим файл message_to_john.asc с содержанием:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hello, John!
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAEBAgAGBQJVY1r3AAoJEI5UpplI5ZOGoWwIAObDda3SfJJ2OWr/6qsPbEs3
CP+WzDiBbAc1wFrByTpuDLt4dxatQKeY1O8EzF4xOJnKrjo9JoQqrrH5DyMq10NO
L9Oo3Lqq6xTcf1vhPusGPHo8qT9rl7bSPiDkQ9Xhov/5BzTeCvuin1h6PVXKn07z
AJYF/8FUSJJpD6EPVyQ4hcUDnSg2K/nJsGJpxdaHzJ49koLnNro1rSkOew0l3h3Q
P/4jiW48XTKtJW9MY6NclyCZpYJalnLhTTG6/3MtnNk53lffn/qpCNH/Q+NxAjLz
wqfOjyfYpSUk3evP/KmhZ3qphoDH95S9oGzfRzHVnogrm1WGuBGA2OQxR71hl/g=
=FPDa
-----END PGP SIGNATURE-----
```

Теперь, если его расшифровать получим такой же файл как и был(см. выше). Если файл верифицировать, то можно увидеть слеующую информацию:

```
gpg --verify message_to_john.asc

gpg: Подпись создана 05/25/15 20:25:11 RTZ 2 (чшьр) ключом RSA с ID 48E59386
gpg: Действительная подпись от "Bob__ <bob@yandex.ru>" [неизвестной]
gpg: ВНИМАНИЕ: Данный ключ не заверен доверенной подписью!
gpg:           Нет указаний на то, что подпись принадлежит владельцу.
Отпечаток главного ключа: EC71 8B72 E0A5 1345 483A  0639 8E54 A699 48E5 9386
gpg: ВНИМАНИЕ: не отделенная подпись; файл ‘message_to_john’ НЕ был проверен!
```

Так же можно сделать отдельную подпись следующим образом:

```
root@ubuntu:~/Загрузки$ gpg --detach-sign -a message_to_john

Необходим пароль для доступа к секретному ключу пользователя: "Bob__ <bob@yandex.ru>"
2048-бит RSA ключ, ID 48E59386, создан 2015-05-13
```

Теперь отправим файлы на основную машину и верефицируем. Получим следующее сообщение:

```
gpg: предполагается, что подписанные данные находятся в ‘message_to_john’
gpg: Подпись создана 05/25/15 20:40:23 RTZ 2 (чшьр) ключом RSA с ID 48E59386
gpg: Действительная подпись от "Bob__ <bob@yandex.ru>" [неизвестной]
gpg: ВНИМАНИЕ: Данный ключ не заверен доверенной подписью!
gpg:           Нет указаний на то, что подпись принадлежит владельцу.
Отпечаток главного ключа: EC71 8B72 E0A5 1345 483A  0639 8E54 A699 48E5 9386
```

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. Ранее не имев дела с подобными программами был впечатлен простотой подписей и работой с ними. Следует отметить, что программа позволяет использовать сервера с ключами, что тоже удобно. Надеюсь в будущем полученный опыт пригодится.