

# **Лабораторная работа №1**

**Научное программирование**

Колчева Юлия Вячеславовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>12</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>13</b>
<b>6</b>	<b>Список литературы</b>	<b>17</b>

## List of Tables

# List of Figures

3.1	Учетная запись . . . . .	7
3.2	Идентификация и создание ключа . . . . .	8
3.3	Создание ключа . . . . .	8
3.4	Создание ключа pgr . . . . .	9
3.5	Настройка ключа . . . . .	9
3.6	Список и отпечаток . . . . .	9
3.7	Демонстрация ключа . . . . .	10
3.8	Отображение ключа . . . . .	10
3.9	Отображение ключа . . . . .	11
3.10	Отображение ключа . . . . .	11

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

## 2 Задание

Создать базовую конфигурацию для работы с git. 1. Создать ключ SSH. 2. Создать ключ PGP. 3. Настроить подписи git. 4. Зарегистрироваться на Github. 5. Создать локальный каталог для выполнения заданий по предмету.

### 3 Выполнение лабораторной работы

Создаём учётную запись на <https://github.com>. (рис. 3.1 )

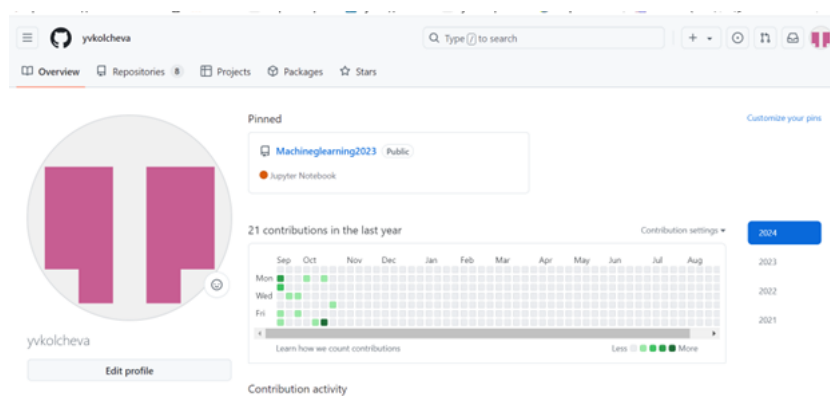


Figure 3.1: Учетная запись

Настроим систему контроля версий git, как это описано в лабораторной работе с использованием сервера репозитория <https://github.com/> 1) Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` (рис. 3.2 ) Настроим верификацию и подписание коммитов git. Зададим имя начальной ветки. 2) Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняться в каталоге `~/.ssh/`. Генерируем ключ двумя способами (рис. 3.3 )

```

yvkolcheva@yvkolcheva:~$ git config --global user.name "yvkolcheva"
yvkolcheva@yvkolcheva:~$ git config --global user.email "kolcheva2001@bk.ru"
yvkolcheva@yvkolcheva:~$ git config --global core.quotepath false
yvkolcheva@yvkolcheva:~$ git config --global init.defaultBranch master
yvkolcheva@yvkolcheva:~$ git config --global core.autocrlf input
yvkolcheva@yvkolcheva:~$ git config --global core.safecrlf warn
yvkolcheva@yvkolcheva:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/yvkolcheva/.ssh/id_rsa): SSH-key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in SSH-key.
Your public key has been saved in SSH-key.pub.
The key fingerprint is:
SHA256:dSp7uavln6dri6bKuHfLpaNtMPIKUnmX3uNKIAgmnk yvkolcheva@yvkolcheva.myguest.virtualbox.org
The key's randomart image is:
+-----+
|         |
|         |
| oo. . S . |
| + E. + . o + |
| . + = *.O + . |
| + B.XoO+.... |
+-----+

```

Figure 3.2: Идентификация и создание ключа

```

yvkolcheva@yvkolcheva:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/yvkolcheva/.ssh/id_ed25519): SSH-key
SSH-key already exists.
Overwrite (y/n)? n
yvkolcheva@yvkolcheva:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/yvkolcheva/.ssh/id_ed25519): ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh.
Your public key has been saved in ssh.pub.
The key fingerprint is:
SHA256:qgonFeyYMzlsQsN1fwBFmVHh1JRJe9WbEWmDh1jg18g yvkolcheva@yvkolcheva.myguest.virtualbox.org
The key's randomart image is:
+--[ED25519 256]--+
| ..+==+ooEB0= |
| ... ..B..+ooo* |
| +o o.o . o = |
| o.. . o |
| Ooo S |
| o= . |
+-----+

```

Figure 3.3: Создание ключа

Генерируем ключ pqr командой `gpg --full-generate-key` (рис. 3.4) и следуя инструкциям из лабораторной работы создаём ключ (рис. 3.5)



```
yvkolcheva@yvkolcheva:~$ gpg --full-generate-key
gpg: команда не найдена...
yvkolcheva@yvkolcheva:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/home/yvkolcheva/.gnupg'
gpg: создан шит с ключами '/home/yvkolcheva/.gnupg/pubring.kbx'
Выберите тип ключа:
(1) RSA и RSA (по умолчанию)
(2) DSA и Elgamal
(3) DSA (только для подписи)
(4) RSA (только для подписи)
(14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (2048) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
0 = не ограничен
<п> = срок действия ключа - п дней
<п>w = срок действия ключа - п недель
<п>m = срок действия ключа - п месяцев
<п>y = срок действия ключа - п лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: yvkolcheva
Адрес электронной почты: kolcheva2001@bk.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"yvkolcheva <kolcheva2001@bk.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
Необходимо получить много случайных чисел. Желательно, чтобы Вы
```

Figure 3.4: Создание ключа gpg

```
yvkolcheva@yvkolcheva:~$ gpg --full-generate-key
(14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (2048) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
0 = не ограничен
<п> = срок действия ключа - п дней
<п>w = срок действия ключа - п недель
<п>m = срок действия ключа - п месяцев
<п>y = срок действия ключа - п лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: yvkolcheva
Адрес электронной почты: kolcheva2001@bk.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"yvkolcheva <kolcheva2001@bk.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
Необходимо получить много случайных чисел. Желательно, чтобы Вы
```

Figure 3.5: Настройка ключа

Выводим список ключей и копируем отпечаток приватного ключа (рис. 3.6 )

```
yvkolcheva@yvkolcheva:~$ gpg --list-secret-keys --keyid-format LONG
gpg: в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/yvkolcheva/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ 523CF256A89935C9 помечен как абсолютно доверенный
gpg: создан каталог '/home/yvkolcheva/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/yvkolcheva/.gnupg/openpgp-revocs.d/113558C39C9F8B7BAF1311FC523CF256A89
935C9.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-09-12 [SC]
      113558C39C9F8B7BAF1311FC523CF256A89935C9
uid          yvkolcheva <kolcheva2001@bk.ru>
sub   rsa4096 2024-09-12 [E]

[yvkolcheva@yvkolcheva ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 доверенных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/home/yvkolcheva/.gnupg/pubring.kbx
-----
sec   rsa4096/523CF256A89935C9 2024-09-12 [SC]
      113558C39C9F8B7BAF1311FC523CF256A89935C9
uid          [ absolutely ] yvkolcheva <kolcheva2001@bk.ru>
ssb   rsa4096/8BB20EAC31F58EFC 2024-09-12 [E]
```

Figure 3.6: Список и отпечаток

Скопируем наш сгенерированный PGP ключ в буфер обмена (рис. 3.7 )

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
  
mQINBGbi2AQBEADfYvv/BcvT3uZB4ieQxGV3hNnrek3ExkTjlcRE5tgMscE2K3D4  
16b2w5aoc1m7K/lrvzDU3NdIQ+w4bRrpMrHNu7vA1Vk7vn/hPYLi2HD025R7LzJX  
PTk+TznPI50c2/bMY2Yn5FjIfVW0nH+l6tqKbJitR9uLWxfCdbwnT3iLY5awxn5a  
ieUv0kyZoKWcu0uCyPsHIDwoQG1hfn74nirkDRJ3C8JWJW0xxhrNB3xLYuIP9v3C  
QbLAtefw4Vv7/ir5Cn4UdKzSBS1eUx2sVpTZgkAPspZygi79rv+7Yxdq5oFvZwU  
TmSr5mvio+eoM6SxaWUwqwpRDof4AFiT5fhSv5llHNNppYFhw+Hmh2DIbVF2XZ3B  
Uz57gNsQGuCU+PK5gjq7eQ80/xSM0eWzX3Z9L7c70dRbXe7HToksiq4wcHC3lxa  
kEyoKvHYGswxi/0hfb9sH9cza1zf/XASyF+bGAG7zpNCLeoFqbQkAyPJq230kSNq  
YqBny6S1ekMwK0uGItk4smfoaEvghaGKMI8bfZ/Lkwt598bGIwGpIRU7z0ns1lxi  
VW7ILSaWw95YbCkA+ZNLyZshaw2JuiXb9LIdXUnSsH0d4MAugApHyliD0PwrU79r  
crIiXBAd75izLE6ijB/Uhhduq0NJoWwKJIm3WijJ+1LJplanz+4snuKb0QARAQAB  
tB95dmtvbGNoZXZhIDxrb2xjaGV2YTIwMDFAYmsucnU+iQJ0BBMBCAA4FiEEETVY  
w5yfi3uvEXH8UjzyVqiZNckFAmbi2AQCGwMFCwkIBwIGFQoJCAAsCBBCAwEChqEC  
F4AACGk0UjzyVqiZNcnYeg/+KIBqmkim3BnF5JhUbjHjGwG8zEO+G06YLHtMieUX  
gAeLfgpv94EigU0C52yR3KHENhYB9vaPfEfVgz8rS2YTICFoISG+joPajzp763X  
v+cmzT6QeaBgqpkE8YXyF9sJ68fiKFE5MEGfPzbFalgyBGzp2ThduaRg150JLfsI  
sunJlCaNlRldyPz46LIzn1nPpfV+0o3rIEZhZj/Njr4HdaDd0RssJI8c/wQwZc/G
```

Figure 3.7: Демонстрация ключа

Перейдем в настройки GitHub (<https://github.com/settings/keys>), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода(рис. 3.8 )

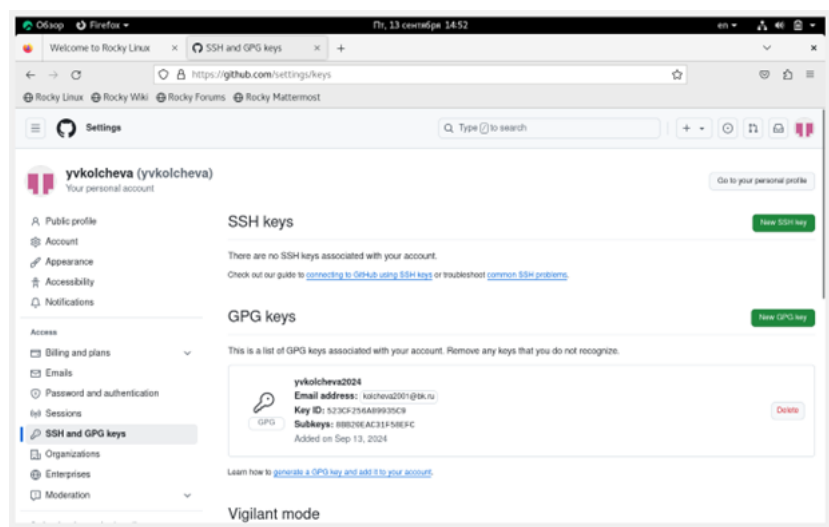
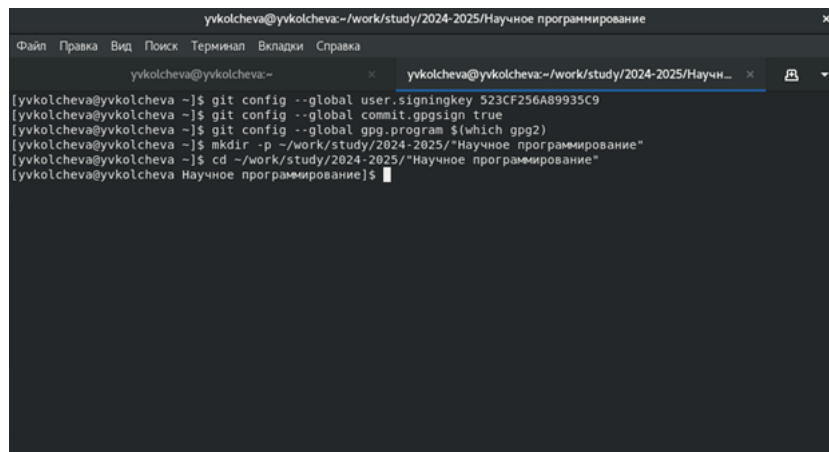


Figure 3.8: Отображение ключа

Осталось настроить автоматические подписи с помощью команд: (рис. 3.9 )



```
yvkolcheva@yvkolcheva:~/work/study/2024-2025/Научное программирование
Файл  Правка  Вид  Поиск  Терминал  Вкладки  Справка
yvkolcheva@yvkolcheva:~
[yvkolcheva@yvkolcheva ~]$ git config --global user.signingkey 523CF256A89935C9
[yvkolcheva@yvkolcheva ~]$ git config --global commit.gpgsign true
[yvkolcheva@yvkolcheva ~]$ git config --global gpg.program $(which gpg2)
[yvkolcheva@yvkolcheva ~]$ mkdir -p ~/work/study/2024-2025/"Научное программирование"
[yvkolcheva@yvkolcheva ~]$ cd ~/work/study/2024-2025/"Научное программирование"
[yvkolcheva@yvkolcheva Научное программирование]$
```

Figure 3.9: Отображение ключа

Далее мы создаём репозиторий на основе шаблона и отправляем все имеющиеся данные на GitHub. (рис. 3.10)

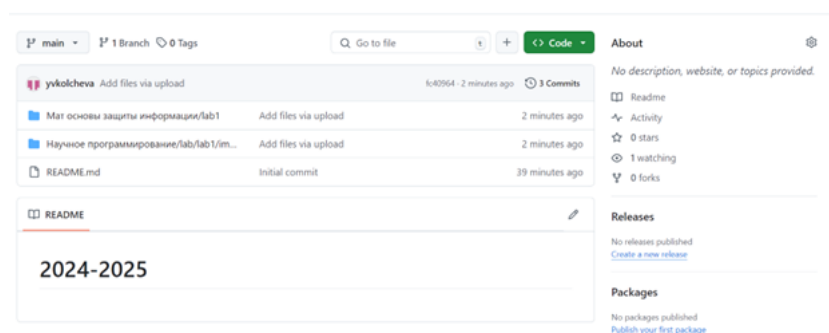


Figure 3.10: Отображение ключа

## 4 Выводы

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий, а также освоила умения по работе с git.

## 5 Ответы на контрольные вопросы

1) Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

2) В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять неполную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

3) Среди классических VCS наиболее известны CVS, Subversion, а среди

распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

4) Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений: `git config --global quotePath false`. Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`. После это в каталог `etutorial` появится каталог `.git`, в котором будет храниться история изменений. Создадим тестовый текстовый файл `hello.txt` и добавим его в локальный репозиторий. `echo 'hello world' > hello.txt git add hello.txt git commit -am 'Новый файл'` Воспользуемся командой `status` для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии: `git status`. 5) Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"`. Ключи сохранятся в каталоге `~/.ssh/`. Существует несколько доступных серверов репозитория с возможностью бесплатного размещения данных. Например, <https://github.com/>. Для работы с ним необходимо сначала зайти на сайт <https://github.com/> и создать учётную запись. Затем необходимо загрузить сгенерённый нами ранее открытый ключ. Для этого зайти на сайт <https://github.com/> под своей учётной записью и перейти в меню `GitHub setting`. После этого выбрать в боковом меню `GitHub setting` SSH-ключи и нажать кнопку `Добавить ключ`. Скопировав из локальной консоли ключ в буфер обмена: `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле. После этого можно создать на сайте репозиторий, выбрав в меню `Репозитории` `Создать репозиторий`, дать ему название и сделать общедоступным (публичным). Для загрузки репозитория из локального каталога на сервер выполняем следующие команды: `git remote add origin ssh://git@github.com/.git` `git push -u origin master`. Далее на локальном компьютере можно выполнять стандартные процедуры для работы с git при наличии

центрального репозитория.

6) У Git две основных задачи: первая — хранить информацию о всех изменениях в коде, а вторая — обеспечение удобства командной работы над кодом.

7) Наиболее часто используемые команды git: — создание основного дерева репозитория: `git init` — получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` — отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` — просмотр списка изменённых файлов в текущей директории: `git status` — просмотр текущих изменений: `git diff` — сохранение текущих изменений: — добавить все изменённые и/или созданные файлы и/или каталоги: `git add .` — добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` — удалить файлы/или каталоги из индекса репозитория (при этом файлы/или каталог остаётся в локальной директории): `git rm имена_файлов` — сохранение добавленных изменений: — сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'` — сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit` — создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` — переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) — отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки` — слияние ветки стекущим деревом: `git merge --no-ff имя_ветки` — удаление ветки: — удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки` — принудительное удаление локальной ветки: `git branch -D имя_ветки` — удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8) Гид для создания текстового файла `echo 'hello world' > hello.txt` `git add hello.txt`

9) Для фиксации истории проекта в рамках этого процесса вместо одной ветки `master` используются две ветки. В ветке `master` хранится официальная история релиза, а ветка `develop` предназначена для объединения всех функций. Ветви

решают следующие проблемы нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архивами сложно перетаскивать изменения между архивами легко запутаться в файлах

- 10) Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для C и C++ `curl -L -s https://www.gitignore.io/api/c` » `.gitignore` `curl -L -s https://www.gitignore.io/api/c++` » `.gitignore`.



## 6 Список литературы

Лабораторная работа №1

Лабораторная работа № 2. Управление версиями [Электронный ресурс]. 2019.

URL:<https://esystem.rudn.ru/mod/resource/view.php?id=1154997>