

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей**

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплина: Операционные системы

Студент: Бровкин Александр

Группа: НБИбд-01-21

Ст. билет №: 1032215006

Москва

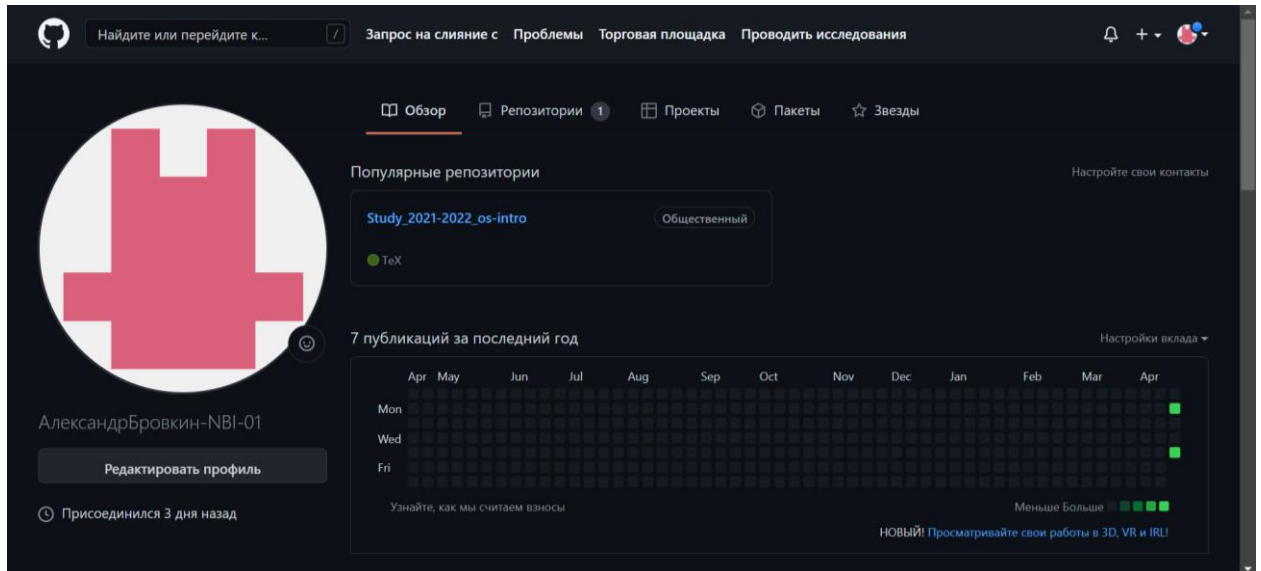
2022 г.

Цель работы

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

Последовательность выполнения работы

Настройка github - Создайте учётную запись на <https://github.com>



У меня уже была учетная запись в гитхаб.

Установка git-flow в Fedora Linux

- Это программное обеспечение удалено из репозитория.
- Необходимо устанавливать его вручную:

```
abrovkin@fedora:~/tmp
abrovkin@fedora ~]$ cd /tmp
[abrovkin@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[abrovkin@fedora tmp]$ chmod +x gitflow-installer.sh
[abrovkin@fedora tmp]$ sudo ./gitflow-installer.sh install stable

Мы полагаем, что ваш системный администратор изложил вам основы
безопасности. Как правило, всё сводится к трём следующим правилам:

#1) Уважайте частную жизнь других.
#2) Думайте, прежде что-то вводить.
#3) С большой властью приходит большая ответственность.

[sudo] пароль для abrovkin:
## git-flow no-make installer ##
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
Клонирование в «gitflow»...
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Получение объектов: 100% (4270/4270), 1.74 MiB | 595.00 КиБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
Переключено на новую ветку «master»
install: создание каталога '/usr/local/share/doc'
install: создание каталога '/usr/local/share/doc/gitflow'
install: создание каталога '/usr/local/share/doc/gitflow/hooks'
'gitflow/git-flow' -> '/usr/local/bin/git-flow'
'gitflow/git-flow-init' -> '/usr/local/bin/git-flow-init'
'gitflow/git-flow-feature' -> '/usr/local/bin/git-flow-feature'
'gitflow/git-flow-bugfix' -> '/usr/local/bin/git-flow-bugfix'
'gitflow/git-flow-hotfix' -> '/usr/local/bin/git-flow-hotfix'
'gitflow/git-flow-release' -> '/usr/local/bin/git-flow-release'
'gitflow/git-flow-support' -> '/usr/local/bin/git-flow-support'
'gitflow/git-flow-version' -> '/usr/local/bin/git-flow-version'
'gitflow/gitflow-common' -> '/usr/local/bin/gitflow-common'
'gitflow/gitflow-shFlags' -> '/usr/local/bin/gitflow-shFlags'
'gitflow/git-flow-config' -> '/usr/local/bin/git-flow-config'
'gitflow/hooks/filter-flow-hotfix-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-finish-tag-message'
'gitflow/hooks/filter-flow-hotfix-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-start-version'
'gitflow/hooks/filter-flow-release-branch-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-branch-tag-message'
'gitflow/hooks/filter-flow-release-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-finish-tag-message'
'gitflow/hooks/filter-flow-release-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-start-version'
'gitflow/hooks/post-flow-bugfix-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-delete'
```

```
Чт, 21 апреля 13:57
abrovkin@fedora:tmp

flow/hooks/filter-flow-hotfix-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-finish-tag-message'
'gitflow/hooks/filter-flow-hotfix-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-start-version'
'gitflow/hooks/filter-flow-release-branch-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-branch-tag-message'
'gitflow/hooks/filter-flow-release-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-finish-tag-message'
'gitflow/hooks/filter-flow-release-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-start-version'
'gitflow/hooks/post-flow-bugfix-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-delete'
'gitflow/hooks/post-flow-bugfix-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-finish'
'gitflow/hooks/post-flow-bugfix-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-publish'
'gitflow/hooks/post-flow-bugfix-pull' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-pull'
'gitflow/hooks/post-flow-bugfix-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-start'
'gitflow/hooks/post-flow-bugfix-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-track'
'gitflow/hooks/post-flow-feature-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-delete'
'gitflow/hooks/post-flow-feature-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-finish'
'gitflow/hooks/post-flow-feature-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-publish'
'gitflow/hooks/post-flow-feature-pull' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-pull'
'gitflow/hooks/post-flow-feature-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-start'
'gitflow/hooks/post-flow-feature-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-track'
'gitflow/hooks/post-flow-hotfix-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-delete'
'gitflow/hooks/post-flow-hotfix-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-finish'
'gitflow/hooks/post-flow-hotfix-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-publish'
'gitflow/hooks/post-flow-hotfix-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-start'
'gitflow/hooks/post-flow-release-branch' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-branch'
'gitflow/hooks/post-flow-release-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-delete'
'gitflow/hooks/post-flow-release-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-publish'
'gitflow/hooks/post-flow-release-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-start'
'gitflow/hooks/post-flow-release-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-track'
'gitflow/hooks/pre-flow-feature-delete' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-delete'
'gitflow/hooks/pre-flow-feature-finish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-finish'
'gitflow/hooks/pre-flow-feature-publish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-publish'
'gitflow/hooks/pre-flow-feature-pull' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-pull'
'gitflow/hooks/pre-flow-feature-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-start'
'gitflow/hooks/pre-flow-feature-track' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-track'
'gitflow/hooks/pre-flow-hotfix-delete' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-delete'
'gitflow/hooks/pre-flow-hotfix-finish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-finish'
'gitflow/hooks/pre-flow-hotfix-publish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-publish'
'gitflow/hooks/pre-flow-hotfix-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-start'
'gitflow/hooks/pre-flow-release-branch' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-branch'
'gitflow/hooks/pre-flow-release-delete' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-delete'
'gitflow/hooks/pre-flow-release-finish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-finish'
'gitflow/hooks/pre-flow-release-publish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-publish'
'gitflow/hooks/pre-flow-release-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-start'
'gitflow/hooks/pre-flow-release-track' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-track'
abrovkin@fedora:tmp$
```

Установка gh в Fedora Linux

```
abrovkin [Рабочий] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Обзор  Терминал  Чт, 21 апреля 14:01  en  [иконки]
abrovkin@fedora:tmp

[abrovkin@fedora tmp]$ sudo dnf install gh
Fedora 35 - x86_64 - Updates                               1.7 kB/s | 7.2 kB   00:04
Fedora 35 - x86_64 - Updates                               1.3 MB/s | 3.2 MB   00:02
Fedora Modular 35 - x86_64 - Updates                      39 kB/s | 19 kB   00:00
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий      Размер
-----
Установка: gh-2.7.0-1.fc35
Установка 1 Пакет
=====
Результат транзакции
=====
Установка 1 Пакет
=====
Объем загрузки: 6.8 М
Объем изменений: 32 М
Продолжить? [Д/Н]: Д
Загрузка пакетов:
gh-2.7.0-1.fc35.x86_64.rpm                2.4 MB/s | 6.8 MB   00:02
-----
Общий размер                               2.0 MB/s | 6.8 MB   00:03
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка :
Установка : gh-2.7.0-1.fc35.x86_64 1/1
Запуск скриптов: gh-2.7.0-1.fc35.x86_64 1/1
Проверка : gh-2.7.0-1.fc35.x86_64 1/1
Установлен:
gh-2.7.0-1.fc35.x86_64
Выполнено!
```

Базовая настройка git

Задам имя и email владельца репозитория:

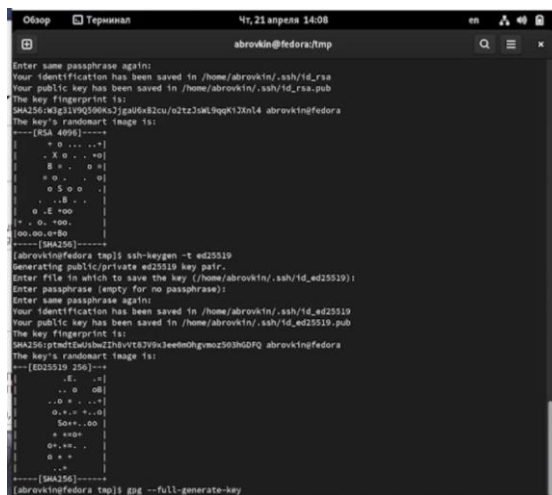
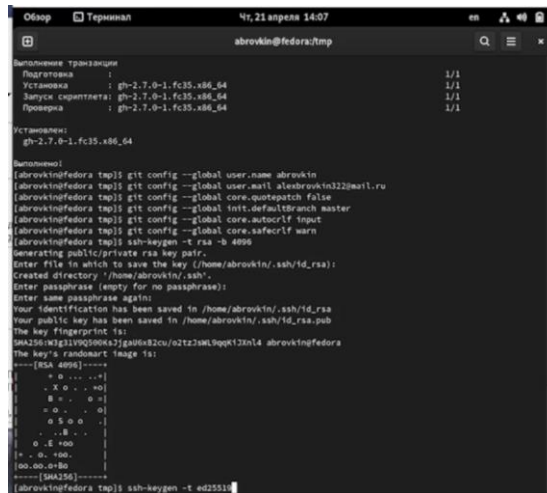
Настрою верификацию и подписание коммитов git. – Задаю имя начальной ветки (будем называть её master):

Параметр autocrlf:

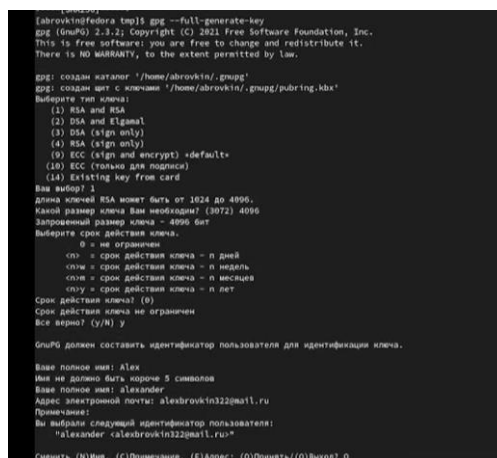
Параметр safecrlf:

```
[abrovkin@fedora tmp]$ git config --global user.name abrovkin
[abrovkin@fedora tmp]$ git config --global user.mail alexbrovkin322@mail.ru
[abrovkin@fedora tmp]$ git config --global core.quotepatch false
[abrovkin@fedora tmp]$ git config --global init.defaultBranch master
[abrovkin@fedora tmp]$ git config --global core.autocrlf input
[abrovkin@fedora tmp]$ git config --global core.safecrlf warn
```

И



Создаю ключи pgr – Генерируем ключ



```
abrovkin [работает] - Oracle VM VirtualBox
Файл Машина Вид Вид Устройства Справка
Обзор Терминал Чт, 21 апреля 14:12 en
abrovkin@fedora:tmp

<ли> = срок действия ключа - п недели
<ли> = срок действия ключа - п месяцев
<ли> = срок действия ключа - п лет
Срок действия ключа? (0)
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Alex
Имя не должно быть короче 5 символов
Ваше полное имя: alexander
Адрес электронной почты: alexbrovkin322@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "alexander <alexbrovkin322@mail.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес, (O)Почты, (Q)Выйти? 0
Необходимо получить много случайных чисел. К сожалению, чтобы вы
в процессе генерации выполняли какие-то другие действия (печатать
на клавиатуре, движение мыши, обращение к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. К сожалению, чтобы вы
в процессе генерации выполняли какие-то другие действия (печатать
на клавиатуре, движение мыши, обращение к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/abrovkin/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ 9680C05786F177C6 помечен как абсолютно доверенный
gpg: создан каталог '/home/abrovkin/.gnupg/openpgp-revocs.d'
gpg: сертификат отныне записан в '/home/abrovkin/.gnupg/openpgp-revocs.d/37516034841622A559A09FA9680C05786F177C6.rev'
Открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-21 [SC]
      37516034841622A559A09FA9680C05786F177C6
uid    alexander <alexbrovkin322@mail.ru>
sub    rsa4096 2022-04-21 [E]

[abrovkin@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
```

Добавление PGP ключа в GitHub

- Выводим список ключей и копируем отпечаток приватного ключа: `gpg --list-secret-keys --keyid-format LONG`
- Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.

```
abrovkin@fedora:tmp

pub   rsa4096 2022-04-21 [SC]
      37516034841622A559A09FA9680C05786F177C6
uid    alexander <alexbrovkin322@mail.ru>
sub    rsa4096 2022-04-21 [E]

[abrovkin@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: про세스가 실패했습니다
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: keybox: 0 signatures: 1 main: 0
/home/abrovkin/.gnupg/pubring.kbx

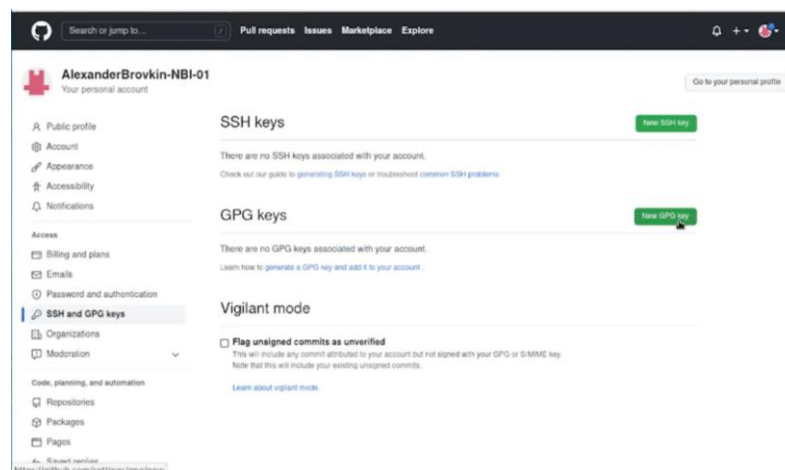
sec   rsa4096/9680C05786F177C6 2022-04-21 [SC]
      37516034841622A559A09FA9680C05786F177C6
uid    [ alexmemo ] alexander <alexbrovkin322@mail.ru>
sub    rsa4096/992F63395E59972 2022-04-21 [E]

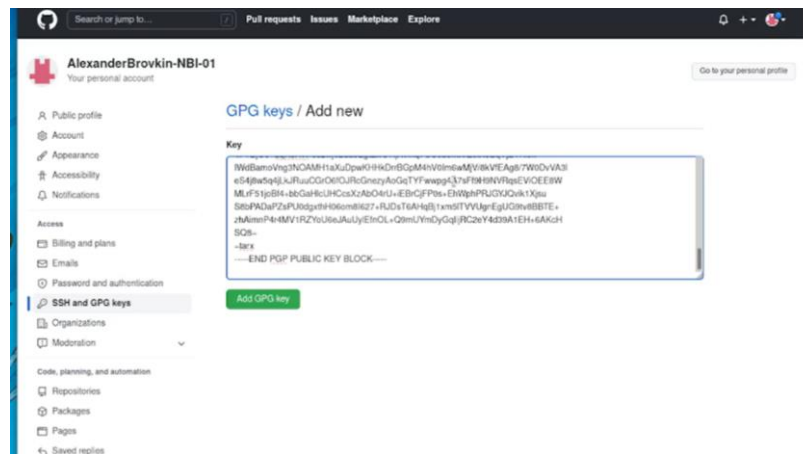
[abrovkin@fedora tmp]$ ^C
[abrovkin@fedora tmp]$ gpg --armor --export 'C
[abrovkin@fedora tmp]$ gpg --armor --export 9680C05786F177C6 | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y] y

• Waiting in queue...
The following packages have to be installed:
xclip-1:15-11.1.fc35.x86_64 Command line clipboard grabber
Proceed with changes? [N/y] y

• Waiting in queue...
• Waiting for authentication...
• Waiting in queue...
• Downloading packages...
• Requesting data...
• Testing changes...
• Installing packages...

[abrovkin@fedora tmp]$ gpg --armor --export 9680C05786F177C6 | xclip -sel clip
[abrovkin@fedora tmp]$
```





Настройка автоматических подписей коммитов git
– Используя введенный email, укажу Git применять его при подписи коммитов:

```
abrovkin@fedora:tmp
375116034841622A55A99FA9680C05786F177C6
alexander.salexbrovkin322@mail.ru
rsa4096 2022-04-21 [E]

[abrovkin@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: пропущен файл joseph
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: keybox: 0 достояние: 1 подписанная: 0 доверие: 0, 0a, 0b, 0f, 1a
/home/abrovkin/.gnupg/pubring.kbx

sec rsa4096/9680C05786F177C6 2022-04-21 [SC]
375116034841622A55A99FA9680C05786F177C6
[ alexandro ] alexander.salexbrovkin322@mail.ru
rsa4096/992F83385E659972 2022-04-21 [E]

[abrovkin@fedora tmp]$ ^C
[abrovkin@fedora tmp]$ gpg --armor --export ^C
[abrovkin@fedora tmp]$ gpg --armor --export 9680C05786F177C6 | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y] y

* Waiting in queue...
the following packages have to be installed:
xclip-0.13-15.git11cb061.fc35.x86_64 Command line clipboard grabber
Proceed with changes? [N/y] y

* Waiting in queue...
* Waiting for authentication...
* Waiting in queue...
* Downloading packages...
* Requesting data...
* Testing changes...
* Installing packages...

[abrovkin@fedora tmp]$ gpg --armor --export 9680C05786F177C6 | xclip -sel clip
[abrovkin@fedora tmp]$ git config --global user.signingkey 9680C05786F177C6
[abrovkin@fedora tmp]$ git config --global user.email alexandro@fedora.ru
```

```
abrovkin@fedora:tmp -- gh auth login

--rename-section переименовать раздел: старое-имя новое-имя
--remove-section удалить раздел: имя
-l, --list показать весь список
--fixed-value use string equality when comparing values to 'value-pattern'
-s, --edit открыть в редакторе
--get-color найди настроенный цвет: раздел [no-упоминание]
--get-colorbool проверить, существует ли настроенный user: раздел [stdout-есть-ty]

Tm
-t, --type <v> value is given this type
--bool значение - это true (правда) или <false> (ложь)
--int значение - это десятичное число
--bool-or-int значение - это --bool или --int
--bool-or-str value is --bool or string
--path значение - это путь (к файлу или каталогу)
--expiry-date значение - это дата окончания срока действия

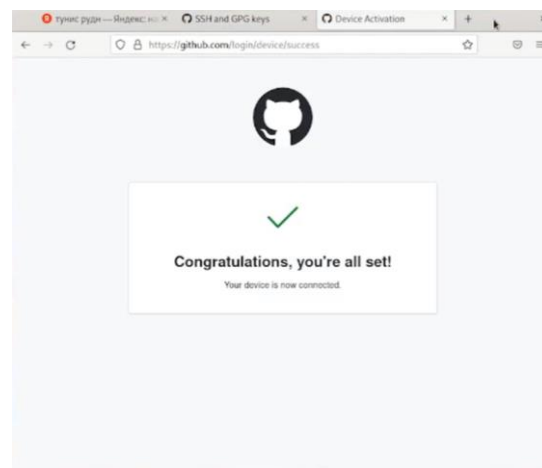
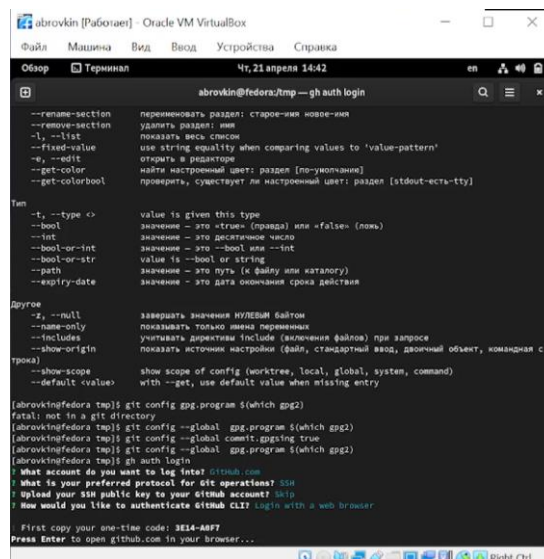
Zyros
-z, --null завершать значения нулевым байтом
--name-only показывать только имена переменных
--includes учитывать директивы include (включения файлов) при запросе
--show-origin показать источник настроек (файл, стандартный ввод, двоичный объект, команда и т.д.)
--show-scope show scope of config (worktree, local, global, system, command)
--default <value> with --get, use default value when missing entry

[abrovkin@fedora tmp]$ git config gpg.program $(which gpg)
fatal: not in a git directory
[abrovkin@fedora tmp]$ git config --global gpg.program $(which gpg)
[abrovkin@fedora tmp]$ git config --global commit.gpgsign true
[abrovkin@fedora tmp]$ git config --global gpg.program $(which gpg)
[abrovkin@fedora tmp]$ gh auth login

What account do you want to log into? GitHub.com
What is your preferred protocol for git operations? ssh
Upload your SSH public key to your GitHub account? skip
How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: 3E34-AB07
Press Enter to open github.com in your browser...
```


Настройка gh – Для начала необходимо авторизоваться gh auth login
– Утилита задаст несколько наводящих вопросов. – Авторизоваться можно через браузер.



Создание репозитория курса на основе шаблона

Настройка каталога курса
Перейдите в каталог курса:
Удалю лишние файлы:
Создам необходимые каталоги:
Отправлю файлы на сервер:

```
[abrovkin@fedora tmp]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
```

```
[abrovkin@fedora tmp]$ cd ~/work/study/2021-2022/"Операционные системы"
[abrovkin@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro
--template=yamadharma/course-directory-student-template --public
Welcome to GitHub CLI!
```


```
✓ Logged in as AlexanderBrovkin-NBI-01
[abrovkin@fedora Операционные системы]$ git clone --recursive git@github.com:AlexanderBrovkin-NBI-01/study_2021-2022_os-intro.git os-intro
Клонирование в «os-intro»...
```

```
abrovkin@fedora:~/work/study/2021-2022/Операционные системы/os-intro
create mode 100644 project-personal/stage2/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage2/report/report.md
create mode 100644 project-personal/stage3/presentation/Makefile
create mode 100644 project-personal/stage3/presentation/presentation.md
create mode 100644 project-personal/stage3/report/Makefile
create mode 100644 project-personal/stage3/report/bib/cite.bib
create mode 100644 project-personal/stage3/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage3/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage3/report/report.md
create mode 100644 project-personal/stage4/presentation/Makefile
create mode 100644 project-personal/stage4/presentation/presentation.md
create mode 100644 project-personal/stage4/report/Makefile
create mode 100644 project-personal/stage4/report/bib/cite.bib
create mode 100644 project-personal/stage4/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage4/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage4/report/report.md
create mode 100644 project-personal/stage5/presentation/Makefile
create mode 100644 project-personal/stage5/presentation/presentation.md
create mode 100644 project-personal/stage5/report/Makefile
create mode 100644 project-personal/stage5/report/bib/cite.bib
create mode 100644 project-personal/stage5/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage6/report/report.md
create mode 100644 structure
[abrovkin@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 265.87 КиБ | 2.03 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:AlexanderBrovkin-NBI-01/study_2021-2022_os-intro.git
afeb362..97836b3 master -> master
[abrovkin@fedora os-intro]$
```

Вот все последние команды, которые я использовал

```
git add .
git commit -am 'feat(main): make course structure'
git push
```

И вот мой репозиторий

 АлександрБровкин-NBI-01

Добавить файлы через загрузку

авг12д 5 часов назад 4 фиксации

конфигурация	Первоначальная фиксация	5 часов назад
лаборатории	Добавить файлы через загрузку	5 часов назад
проектно-личный	feat(main): создать структуру курса	5 часов назад
шаблон	Первоначальная фиксация	5 часов назад
.gitатрибуты	Первоначальная фиксация	5 часов назад
.gitignore	Первоначальная фиксация	5 часов назад
.gitmodules	Первоначальная фиксация	5 часов назад
ЛИЦЕНЗИЯ	Первоначальная фиксация	5 часов назад
Makefile	Первоначальная фиксация	5 часов назад

Нет описания, веб-сайта или тем.

Прочти меня

Лицензия CC-BY-4.0

0 звезд

1 просмотр

0 вилок

Релизы

Выпуски не опубликованы

Создать новый выпуск

Пакеты

Пакеты не опубликованы

АлександрБровкин-NBI-01 Добавить файлы через загрузку		а68c12d 5 часов назад	История
..			
лаборатория01	Добавить файлы через загрузку	5 часов назад	
лаборатория02	feat(main): создать структуру курса	5 часов назад	
лаборатория03	feat(main): создать структуру курса	5 часов назад	
лаборатория04	feat(main): создать структуру курса	5 часов назад	
лаборатория05	feat(main): создать структуру курса	5 часов назад	
лаборатория06	feat(main): создать структуру курса	5 часов назад	
лаборатория07	feat(main): создать структуру курса	5 часов назад	
лаборатория08	feat(main): создать структуру курса	5 часов назад	
лаборатория09	feat(main): создать структуру курса	5 часов назад	

Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git?
7. Назовите и дайте краткую характеристику командам git.
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветви (branches)?
10. Как и зачем можно игнорировать некоторые файлы при commit?

Ответы:

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Система контроля версий (VCS) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Для примеров в этой книге мы будем использовать исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа. Если вы графический или веб-дизайнер и хотели бы хранить каждую версию изображения или макета — а этого вам наверняка хочется — то пользоваться системой контроля версий будет очень мудрым решением. даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы за всё, что вы получаете, будут очень маленькими.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище-система, которая обеспечивает хранение всех существовавших вариантов файлов
Commit-фиксация изменений
История-список предыдущих ревизий
Рабочая копия-копия другой ветки
Команде commit можно передать сообщение, описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время

и временную зону, плюс список измененных файлов и их содержимого. Сообщение, описывающее изменения, определяется через опцию -m, или – message. Можно также вводить сообщения, состоящие из нескольких строк; в большинстве оболочек вы можете сделать это оставив открытую кавычку в конце строки. `commit -m "добавлен первый файл.`

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Системы контроля версий. Централизованная система контроля версий Subversion и децентрализованная система контроля версий Mercurial. Существуют СКВ централизованные, в которых имеется один репозиторий, в который собираются изменения со всех рабочих копий разработчиков, и децентрализованные, когда репозитория много, и они могут обмениваться изменениями между собой. Централизованные СКВ - репозиторий один. У каждого разработчика своя рабочая копия. Время от времени разработчик может затягивать к себе в рабочую копию новые изменения из репозитория, или проталкивать свои изменения из своей рабочей копии в репозиторий. Прочие особенности централизованных СКВ зависят от реализации.

4. Опишите действия с VCS при единоличной работе с хранилищем.

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

5. Опишите порядок работы с общим хранилищем VCS.

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельт компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней

версией. Некоторые системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

6. Каковы основные задачи, решаемые инструментальным средством git?

Устанавливает единственную новую команду, git. Все возможности предоставляются через подкоманды этой команды. Вы можете просмотреть краткую справку командой help. Некоторые идеи группируются по темам, используйте help topics для списка доступных тем. Одна из функций системы контроля версий — отслеживать кто сделал изменения. В распределенных системах для этого требуется идентифицировать каждого автора уникально в глобальном плане. Большинство людей уже имеют такой идентификатор: email адрес. Bazaar достаточно умен, чтобы автоматически создавать email адрес из текущего имени и адреса хоста.

Основные задачи: создание ветки, размещение веток, просмотр изменений, фиксация изменений, сообщение из текстового редактора, выборочная фиксация, удаление зафиксированных изменений, игнорирование файлов, просмотр истории, статистика ветки, контроль файлов и каталогов, ветвление, объединение веток, публикация ветки.

7. Назовите и дайте краткую характеристику командам git.

Обновление рабочей копии По мере внесения изменений в проект рабочая копия на компьютере разработчика стареет, расхождение её с основной версией проекта увеличивается. Это повышает риск возникновения конфликтных изменений (см. ниже). Поэтому удобно поддерживать рабочую копию в состоянии, максимально близком к текущей основной версией, для чего разработчик выполняет операцию обновления рабочей копии (update) насколько возможно часто (реальная частота обновлений определяется частотой внесения изменений, зависящей от активности разработки и числа разработчиков, а также временем, затрачиваемым на каждое обновление — если оно велико, разработчик вынужден ограничивать частоту обновлений, чтобы не терять время).
Модификация проекта Разработчик модифицирует проект, изменяя входящие в него файлы в рабочей копии в соответствии с проектным заданием. Эта работа производится локально и не требует обращений к серверу VCS.
Фиксация изменений Завершив очередной этап работы над заданием, разработчик фиксирует (commit) свои изменения, передавая их на сервер (либо в основную ветвь, если работа над заданием полностью завершена, либо в отдельную ветвь разработки данного задания). VCS может требовать от разработчика перед фиксацией обязательно выполнить обновление рабочей копии. При наличии в системе поддержки отложенных изменений (shelving) изменения могут быть переданы на сервер без фиксации. Если утверждённая политика работы в VCS это позволяет, то фиксация изменений может проводиться не ежедневно, а только по завершении работы над заданием; в этом случае до завершения работы все связанные с заданием изменения сохраняются только в локальной рабочей копии разработчика.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Мы создаем новую ветку выполнив git init в уже созданном каталоге: % mkdir tutorial % cd tutorial % ls -a ./ ../ % pwd /home/mbp/work/bzr.test/tutorial % % git init % ls -aF ./ ../ .git/ % Мы обычно обращаемся к веткам на нашем компьютере просто передав имя каталога содержащего ветку. bzr также поддерживает доступ к веткам через http и sftp, например: git log http://bazaar-vcs.org git // git.dev/ git log sftp://bazaarvcs.org/bzr/bzr.dev/ Установив для git плагины можно также осуществлять доступ к веткам с использованием rsync. Команда status показывает какие изменения были сделаны в рабочем каталоге с момента последней ревизии: % git status modified: foo bzr status скрывает неинтересные файлы, которые либо не менялись, либо игнорируются. Также команде status могут быть переданы необязательные имена файлов, или каталогов для проверки. Команда diff

показывает изменения в тексте файлов в стандартном формате diff. Вывод этой команды может быть передан другим командам, таким как "patch", "diffstat", "filterdiff" и "colordiff": % git diff == added file 'hello.txt' --- hello.txt 1970-01-01 00:00:00 +0000 +++ hello.txt 2005-10-18 14:23:29 +00006.2. Указания к лабораторной работе 75 @@ -0,0 +1,1 @@ +hello world Команде commit можно передать сообщение описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. git commit -m "добавлен первый файл" Если вы передадите список имен файлов, или каталогов после команды commit, то будут зафиксированы только изменения для переданных объектов. Например: bzr commit -m "исправления документации" commit.py Если вы сделали какие-либо изменения и не хотите оставлять их, используйте команду revert, что бы вернуться к состоянию предыдущей ревизии. Многие деревья с исходным кодом содержат файлы которые не нужно хранить под контролем версий, например резервные файлы текстового редактора, объектные файлы и собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать git игнорировать их добавив их в файл .ignore в корне рабочего дерева. Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте команду ignored: % ignored config.h ./config.h configure.in~ *~ log Команда bzr log показывает список предыдущих ревизий. Команда log --forward делает тоже самое, но в хронологическом порядке, показывая более поздние ревизии в конце может контролировать файлы и каталоги, отслеживая переименования и упрощая их последующее объединение: % mkdir src % echo 'int main() {}' > src/simple.c % add src added src added src/simple.c % status added: src/ src/simple.c bzr remove удаляет файл из под контроля версий, но может и не удалять рабочую копию файла2. Это удобно, когда вы добавили не тот файл, или решили, что файл на самом деле не должен быть под контролем версий. % rm -r src % remove -v hello.txt ? hello.txt % status removed: hello.txt src/ src/simple.c unknown: hello.txt Часто вместо того что бы начинать свой собственный проект, выхотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной новой веткой. Если две ветки разошлись (обе имеют уникальные изменения) тогда merge — это подходящая команда для использования. Объединение автоматически вычислит изменения, которые существуют на объединяемой ветке и отсутствуют в локальной ветке и попытается объединить их с локальной веткой. git merge URL.

9. Что такое и зачем могут быть нужны ветви (branches)?

Часто вместо того что бы начинать свой собственный проект, вы хотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной новой веткой эта команда называется branch: Управление версиями git branch cd git.dev Эта команда копирует полную историю ветки и после этого вы можете делать все операции с ней локально: просматривать журнал, создавать и объединять другие ветки.

10. Как и зачем можно игнорировать некоторые файлы при commit?

Нет проблем если шаблон для игнорирования подходит для файла под контролем версий, или вы добавили файл, который игнорируется. Шаблоны не имеют никакого эффекта на файлы под контролем версий, они только определяют показываются неизвестные файлы, или просто игнорируются. Файл git.rignore обычно должен быть под контролем версий, что бы новые копии ветки видели такие же шаблоны: git add . gitignore git commit -m "Добавлены шаблоны для игнорирования". Многие деревья с исходным кодом содержат файлы, которые не нужно хранить под контролем версий, например, резервные файлы текстового редактора, объектные файлы и собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать bzr игнорировать их добавив их в файл в корне рабочего дерева. Этот файл содержит список шаблонов файлов, по одному в каждой строчке. Обычное содержимое

может быть таким: *.o *~ *.tmp *.py [со] Если шаблон содержит слеш, то он будет сопоставлен с полным путем начиная от корня рабочего дерева; иначе он сопоставляется только с именем файла. Таким образом пример выше игнорирует файлы с расширением .o во всех подкаталогах, но пример ниже игнорирует только config.h в корне рабочего дерева и HTML файлы в каталоге doc/: ./config.h doc/*.html Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте команду git ignored : \$ git ignored config.h ./config.h configure.in~ *~ \$

Вывод:

Я научился работать с github и создавать в github каталоги и репозитории, освоил основные умения по работе с git.