

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплина: Операционные системы

Студентка: Арсоева Залина
Группа: НБИбд-01-21

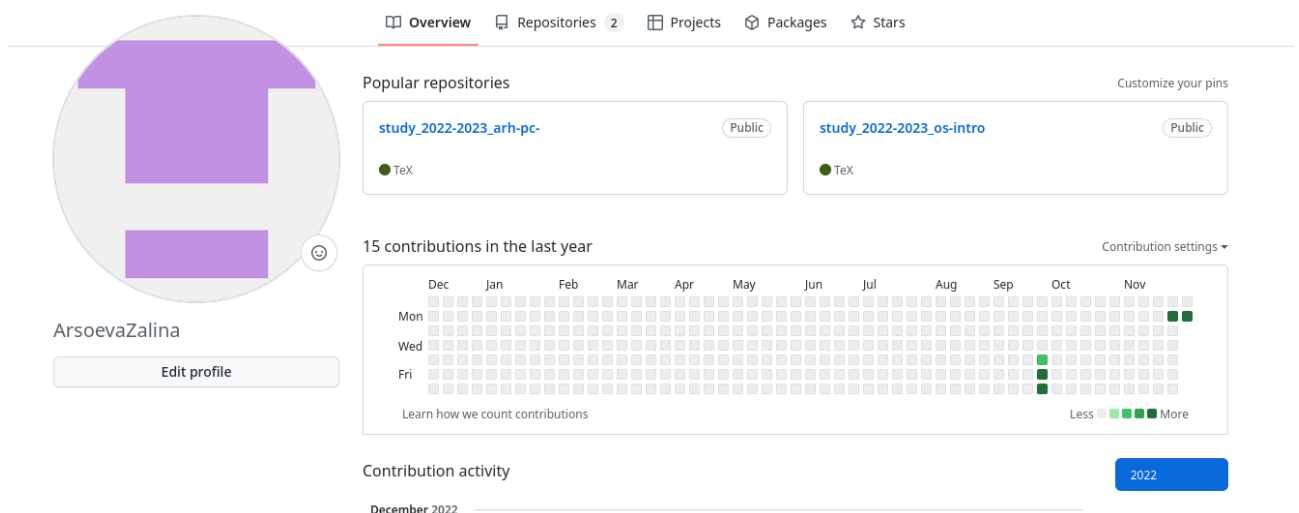
Москва
2022 г.

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:

```
[zalinaarsoeva@fedora ~]$ cd /tmp
[zalinaarsoeva@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
bash: /gitflow/develop/contrib/gitflow-installer.sh: Нет такого файла или каталога
[zalinaarsoeva@fedora tmp]$ chmod +x gitflow-installer.sh
chmod: невозможно получить доступ к 'gitflow-installer.sh': Нет такого файла или каталога
[zalinaarsoeva@fedora tmp]$ wget --no-check-certificate -q http://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[zalinaarsoeva@fedora tmp]$ chmod +x gitflow-installer.sh
[zalinaarsoeva@fedora tmp]$ sudo ./gitflow-installer.sh install stable
[sudo] пароль для zalinaarsoeva:
## 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 МиБ | 2.28 МиБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
branch 'master' set up to track 'origin/master'.
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'
'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'
```

Установка gh в Fedora Linux

```
[zalinaarsoeva@fedora tmp]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:32:50 назад, Пн 05 дек 2022 14:45:37.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.20.2-2.fc36  updates      8.0 М
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 8.0 М
Объем изменений: 41 М
Продолжить? [д/н]: Д
Загрузка пакетов:
gh-2.20.2-2.fc36.x86_64.rpm                2.6 MB/s | 8.0 MB      00:03
=====
Общий размер                               2.1 MB/s | 8.0 MB      00:03
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка      :                               1/1
Установка       : gh-2.20.2-2.fc36.x86_64    1/1
Запуск скрипта : gh-2.20.2-2.fc36.x86_64    1/1
Проверка        : gh-2.20.2-2.fc36.x86_64    1/1
Установлен:
gh-2.20.2-2.fc36.x86_64
```

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:

```
[zalinaarsoeva@fedora tmp]$ git config --global user.name ArsoevaZalina
[zalinaarsoeva@fedora tmp]$ git config --global user.email zalinaarsoeva@yandex.ru
[zalinaarsoeva@fedora tmp]$ git config --global core.quotepath false
[zalinaarsoeva@fedora tmp]$ git config --global init.defaultBranch master
[zalinaarsoeva@fedora tmp]$ git config --global core.autocrlf input
[zalinaarsoeva@fedora tmp]$ git config --global core.safecrlf warn
```

Создаю ключи ssh – по алгоритму rsa с ключём размером 4096 бит:

и

по алгоритму ed25519:

```
[zalinaarsoeva@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zalinaarsoeva/.ssh/id_rsa): 11
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in 11
Your public key has been saved in 11.pub
The key fingerprint is:
SHA256:UZm8plfkgzL1IkCxSgpUFDWJfjQu1fXVD6qbecjeI4 zalinaarsoeva@fedora
The key's randomart image is:
+----[RSA 4096]-----+
| .o+B*+. ..o .. |
| . + oo. .= ... |
| . + o... *o o |
| . o + +o=... . |
| . . . S*.o o |
| . . . . o |
| . . . .o . |
| . . . .ooo. |
| . . . .Eo.... |
+----[SHA256]-----+
```

```
[zalinaarsoeva@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/zalinaarsoeva/.ssh/id_ed25519): 12
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in 12
Your public key has been saved in 12.pub
The key fingerprint is:
SHA256:o39G58wwzDT6KPELMxAxQDW0/Uv4MKp4iA7gevDXJa4 zalinaarsoeva@fedora
The key's randomart image is:
+--[ED25519 256]--+
| .o+* |
| * |
| o . |
| . o o |
| . . + S= . |
| + oo*oo* . |
| o= .o=o+ B |
| =+.. +o + + |
| ++..E. .o+ |
+--[SHA256]-----+
```

Создаю ключи pgp – Генерируем ключ

```
[zalinaarsoeva@fedora tmp]$ gpg --full-generate-key
gpg (GnuPG) 2.3.4; Copyright (C) 2021 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/zalinaarsoeva/.gnupg'
gpg: создан шит с ключами '/home/zalinaarsoeva/.gnupg/pubring.kbx'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
  (10) ECC (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? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n 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: Zali
Имя не должно быть короче 5 символов
Ваше полное имя: Zalina
Адрес электронной почты: zaliarsoeva@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Zalina <zaliarsoeva@yandex.ru>"
```

```
[zaliinaarsoeva@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: gpg
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/home/zaliinaarsoeva/.gnupg/pubring.kbx
-----
sec   rsa4096/A311FB33AA9847EC 2022-12-05 [SC]
      C74EF19C66189ADF6E2230FAA311FB33AA9847EC
uid           [ асолютно ] Zalina <zaliarsoeva@yandex.ru>
ssb   rsa4096/F62CEC80F70D9C52 2022-12-05 [E]

[zaliinaarsoeva@fedora tmp]$ gpg --armor --export <PGP Fingerprint> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
[zaliinaarsoeva@fedora tmp]$ gpg --armor --export <PGP Fingerprint> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
[zaliinaarsoeva@fedora tmp]$ xclip -sel clip
rsa4096/F62CEC80F70D9C52[zaliinaarsoeva@fedora tmp]$ ^C
[zaliinaarsoeva@fedora tmp]$ gpg --armor --export
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINB6ON5VQBEADPdYyqzEf3yU8gSsmfUayiyA1KD/M+5zAT64kDGw0oS6391Pr
gyegC0xkXkAVU4ehjiYVRHy4037qL3zzX20+r0hdmvDuRPLcXo6oZ04kQAa070wBN
UKmFQvrpxDwt45uIMiAYP+DUdcnd02S0x5/erIj988VsviAozECN5PJlr7/92T2
NS098r-ijmErHRB+P+m5tnHXfjKjRXW5BDodFcsba1/MvdhaeGfbbWuKeSpYauG9r
ftXz6UkwmjoxX0LEiWh0D46LBbGj0xqGh1tcXk1UvmIs6p9JWEbm926ziVf5PMw7
z018d5bMb0aNI26/qQvtABDS98pafBBhaGNms0Jv+flYmRL0dEJa5P1Mkd8TjaXu
tH1b3S23A/542ScvPfgMYukXGKSbcYnSszVzuJ6EvV1ae+laXFaQ5kRV9cZ7z5ts
VmsKtAUbmQDH8WAB03cH6qmJP9rwwIqvYiYrHLDPaYTeiKmJqXbhtdNpNrFmmUB
vhnFc0/rkYFPndu6C/ZHy0iHF96+7tiJ4BYC1OKDZ0SiAks6RsyMTXZvM0KATUhn
EqLE8k9Qr5847bgIKcQPxtmxqbgqL8S5XPgvztz47dTj0H/8N9Bpo5nvqbb7Hqd
eL5Unhagqs0Wzym4EPjfJN7q105Zi9zTqLABhFEHfPLZ5J+oj46X+Y4FbwARAQAB
tB5aYwpxbmEgPHphbGlnclNvZXZlLmhmRleC5ydT6JALIEEwEIAAwWlQTHtVgc
Zhia324iMPqjEfSzqphH7AUCY43LVAIbAwULCQgHagMiAgEGF0oJCAsCBBYCAwEC
HgcCF4AAcGkQoxH7M6qYR+wFfA/9FnoVuAlpmYkdLRACNhVChDG3hRBRSMbkCa1h
b8oKk6SasyWiRTi6N+T2dnn79uiKUCG45n7comt854F3Wgl1ji/D5g/D2fHA3//P
A9wcZk/KzzYlljFImZhrnJHG3WnpA0ssYSOLstLlyCAYK+jxPidowEmCF9TF8Adt
PhOyFj2EbS6NrWY3NZN0o2Q00AkBxIyzudnpa72pWhvWsuJbdc7vvKapCrEjdyFo
+qRDv6zkH9xqvN8Iyuu38mFuJFoZr5+0FYitXysOXGeh/yvyELGmwV9Q8ar9Qm1y
vPJvUDejSa5rIeaZzsvxiIPSBYjKlQqH5XjRv3826ytMaF1ncLT1IjJde4y7SE8
JlgJkqZMu0sNCh2Fkye6HEb5hRgOxSxupMjDoYk30HeAkhqdKokFzF799b5GdWAF
q2kv3RzVGUB4CtLmx800l0h/K5wz7E10Z61XvEMsXIgaJxx6JYRKwbvVJvdfSguL
XLS+TcnTN6VpaucYroq0bsInt6KMLah5UQI94NYkbuoN8cgxcH0tmY84M007/M
2jgF2ilbp98IPie6M5LVZF/8Gyl06nHP197nkoZLkmgNkJ4APceEc7Z46e93BPf
xqkVwN0FXpYFwq4/ar/ZzdgawY0L3381TboxcC3rnI5MSQ8HNzf0yqTDVxHhu5oS
NP5Myi5Ag0EY43LVAEQAMK/5hSLcnpBmCBHvdcvN6JL/17c+V1cN10Hpe08v9K
eNyxZG0VYA71XYzS0GzW9mLxHTvzbzGKVRWopB10Fw12uGgtT6Yd1411EYnP5TYCK
```


Добавление GPG ключа в GitHub

- Выводим список ключей и копируем отпечаток приватного ключа: `1 gpg --list-secret-keys --keyid-format LONG` – Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.

GPG keys

New GPG key

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.




11

Email address: zaliarsoeva@yandex.ru

Key ID: A311FB33AA9847EC

Subkeys: F62CEC80F70D9C52

Added on Dec 5, 2022



Delete

Learn how to [generate a GPG key and add it to your account](#).

Настройка автоматических подписей коммитов git

- Используя введенный email, укажу Git применять его при подписи коммитов:

```
[zaliinaarsoeva@fedora tmp]$ git config --global user.singingkey
[zaliinaarsoeva@fedora tmp]$ git config --global commit.gpgsing true
[zaliinaarsoeva@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

```

create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab02/report/report.md
create mode 100644 labs/lab03/presentation/Makefile
create mode 100644 labs/lab03/presentation/image/kulyabov.jpg
create mode 100644 labs/lab03/presentation/presentation.md
create mode 100644 labs/lab03/report/Makefile
create mode 100644 labs/lab03/report/bib/cite.bib
create mode 100644 labs/lab03/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab03/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab03/report/report.md
create mode 100644 labs/lab04/presentation/Makefile
create mode 100644 labs/lab04/presentation/image/kulyabov.jpg
create mode 100644 labs/lab04/presentation/presentation.md
create mode 100644 labs/lab04/report/Makefile
create mode 100644 labs/lab04/report/bib/cite.bib
create mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab04/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab04/report/report.md
create mode 100644 labs/lab05/presentation/Makefile
create mode 100644 labs/lab05/presentation/image/kulyabov.jpg
create mode 100644 labs/lab05/presentation/presentation.md
create mode 100644 labs/lab05/report/Makefile
create mode 100644 labs/lab05/report/bib/cite.bib
create mode 100644 labs/lab05/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab05/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab05/report/report.md
create mode 100644 labs/lab06/presentation/Makefile
create mode 100644 labs/lab06/presentation/image/kulyabov.jpg
create mode 100644 labs/lab06/presentation/presentation.md
create mode 100644 labs/lab06/report/Makefile
create mode 100644 labs/lab06/report/bib/cite.bib

```

Вот все последние команды, которые я использовала

```

[zalinaarsoeva@fedora os-intro]$ git add .
[zalinaarsoeva@fedora os-intro]$ git commit -am 'feat(main): make course structure'
>

```

И вот мой репозиторий

ArsoevaZalina / study_2022-2023_os-intro (Public)
generated from yamadharm/course-directory-student-template
Pin
Unwatch 1
Fork

Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings

master
study_2022-2023_os-intro / labs /
Go to file
Add file
...

ArsoevaZalina feat(main): make course structure e35bad9 34 minutes ago History		
..		
lab01	feat(main): make course structure	34 minutes ago
lab02	feat(main): make course structure	34 minutes ago
lab03	feat(main): make course structure	34 minutes ago
lab04	feat(main): make course structure	34 minutes ago
lab05	feat(main): make course structure	34 minutes ago
lab06	feat(main): make course structure	34 minutes ago
lab07	feat(main): make course structure	34 minutes ago
lab08	feat(main): make course structure	34 minutes ago
lab09	feat(main): make course structure	34 minutes ago

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.