

# **Архитектура операционных систем**

**Лабораторная работа №2**

Леснухин Даниил Дмитриевич НПИбд-02-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>20</b>

## Список иллюстраций

4.1	имя и email владельца репозитория . . . . .	8
4.2	Настройка верификации и подписания коммитов git . . . . .	9
4.3	Настройка верификации и подписания коммитов git . . . . .	9
4.4	Настройка верификации и подписания коммитов git . . . . .	10
4.5	Настройка автоматических подписей коммитов git . . . . .	10

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

### 3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

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 не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

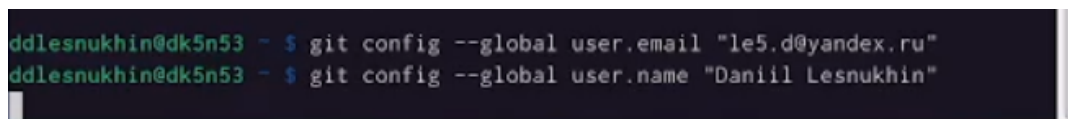
## 4 Выполнение лабораторной работы

1. Зададим имя и email владельца репозитория:

```
git config --global user.name "Name Surname"
```

```
git config --global user.email "work@mail"
```

(рис.[4.1])



```
ddlesnukhin@dk5n53 ~ $ git config --global user.email "le5.d@yandex.ru"
ddlesnukhin@dk5n53 ~ $ git config --global user.name "Daniil Lesnukhin"
```

Рис. 4.1: имя и email владельца репозитория

2. Настраиваем utf-8 в выводе сообщений git:

```
git config --global core.quotePath false
```

(рис. [4.2])

Настройте верификацию и подписание коммитов git (см. Верификация коммитов git с помощью GPG). Зададим имя начальной ветки (будем называть её master):

```
git config --global init.defaultBranch master Параметр autocrlf:
```

```
git config --global core.autocrlf input Параметр safecrlf:
```

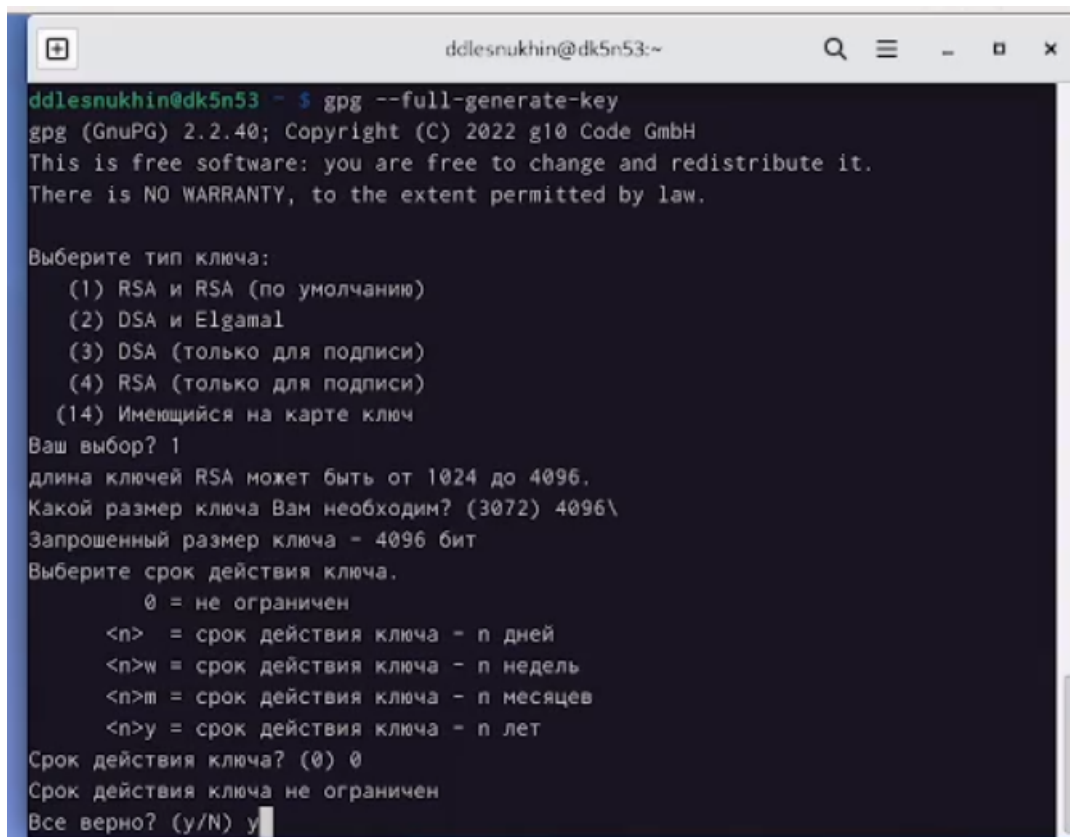
```
git config --global core.safecrlf warn
```



```
ddlesnukhin@dk5n53 ~ $ git config --global core.quotepath false
ddlesnukhin@dk5n53 ~ $ git config --global init.defaultBranch master
ddlesnukhin@dk5n53 ~ $ got config --global core.autocrlf input
bash: got: команда не найдена
ddlesnukhin@dk5n53 ~ $ git config --global core.autocrlf input
ddlesnukhin@dk5n53 ~ $ git config --global core.safecrlf warn
ddlesnukhin@dk5n53 ~ $ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/d/d/ddlesnukhi
n/.ssh/id_rsa): ssh-keygen
```

Рис. 4.2: Настройка верификации и подписания коммитов git

3. Далее создаем ssh и gpg ключи (рис. [4.3]), (рис. [4.4]).



```
ddlesnukhin@dk5n53 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

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

Рис. 4.3: Настройка верификации и подписания коммитов git

```
bash: синтаксическая ошибка рядом с неожиданным маркером «newline»
ddlesnukhin@dk5n53 ~$ gpg --armor --export 089BA2EB29D2F27A | xclip -sel clip
ddlesnukhin@dk5n53 ~$ git config --global user.signingkey 089BA2EB29D2F27A
ddlesnukhin@dk5n53 ~$ git config --global commit.gpgsign true
ddlesnukhin@dk5n53 ~$ git config --global gpg.program $(which gpg2)
ddlesnukhin@dk5n53 ~$
```

Рис. 4.4: Настройка верификации и подписания коммитов git

4. Перейдите в настройки (GitHub)[<https://github.com/settings/keys>], нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода.
5. Настройка автоматических подписей коммитов git Используя введённый email, укажите Git применять его при подписи коммитов:

git config --global user.signingkey (рис. [4.5]) git config --global commit.gpgsign true  
git config --global gpg.program \$(which gpg2)

```
ddlesnukhin@dk5n53 ~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Generate a new SSH key to add to your GitHub account? No
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: A7BD-D7E7
Press Enter to open github.com in your browser...
Окно или вкладка откроется в текущем сеансе браузера.
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Logged in as dalesnoy
ddlesnukhin@dk5n53 ~$
```

Рис. 4.5: Настройка автоматических подписей коммитов git

6. Настройка каталога курса Перейдите в каталог курса:

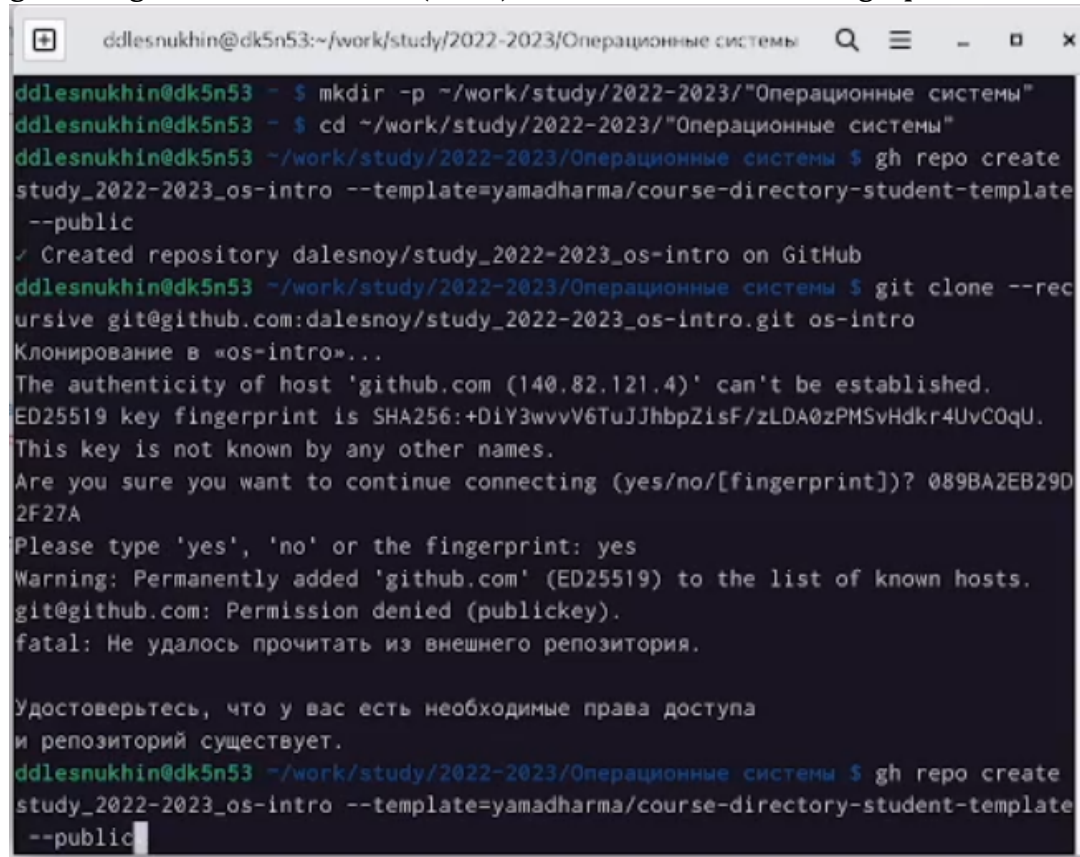
cd ~/work/study/2022-2023/“Операционные системы”/os-intro Удалите лишние файлы:

(рис. [??]), (рис. [??]).

rm package.json Создайте необходимые каталоги:

echo os-intro > COURSE make Отправьте файлы на сервер:

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



```
ddlesnukhin@dk5n53:~/work/study/2022-2023/Операционные системы
ddlesnukhin@dk5n53 ~$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
ddlesnukhin@dk5n53 ~$ cd ~/work/study/2022-2023/"Операционные системы"
ddlesnukhin@dk5n53 ~/work/study/2022-2023/Операционные системы$ gh repo create
study_2022-2023_os-intro --template=yamadharma/course-directory-student-template
--public
✓ Created repository dalesnoy/study_2022-2023_os-intro on GitHub
ddlesnukhin@dk5n53 ~/work/study/2022-2023/Операционные системы$ git clone --rec
ursive git@github.com:dalesnoy/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? 089BA2EB29D
2F27A
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
ddlesnukhin@dk5n53 ~/work/study/2022-2023/Операционные системы$ gh repo create
study_2022-2023_os-intro --template=yamadharma/course-directory-student-template
--public
```

```
ddlesnukhin@dk5n53:~/work/study/2022-2023/Операционные системы
ddlesnukhin@dk5n53 ~ $ mkdir -p ~/work/study/2022-2023/"Операционные системы"
ddlesnukhin@dk5n53 ~ $ cd ~/work/study/2022-2023/Операционные системы
ddlesnukhin@dk5n53 ~/work/study/2022-2023/Операционные системы $ gh repo create
study_2022-2023_os-intro --template=yamadharma/course-directory-student-template
--public
✓ Created repository dalesnoy/study_2022-2023_os-intro on GitHub
ddlesnukhin@dk5n53 ~/work/study/2022-2023/Операционные системы $ git clone --rec
ursive git@github.com:dalesnoy/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? 089BA2EB29D
2F27A
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
ddlesnukhin@dk5n53 ~/work/study/2022-2023/Операционные системы $ gh repo create
study_2022-2023_os-intro --template=yamadharma/course-directory-student-template
--public
```

## 7. Контрольные вопросы

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

Что такое системы контроля версий (VCS) и для решения каких задач они предназначены? 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. 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. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. 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 репозиториев 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. 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.

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. 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.

б) Каковы основные задачи, решаемые инструментальным средством git? 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 распределённых 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. 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. 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 файла<sup>2</sup>. 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)? 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 по-

сле 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? 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 показывающиеся 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 ~ ~ $`

## 5 Выводы

Изучили идеология и применение средств контроля версий. Освоили умения по работе с git.