РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Архитектура компьютера

Студентка: Симбине Камила Шеймиле

Группа: НПИбд-03-23

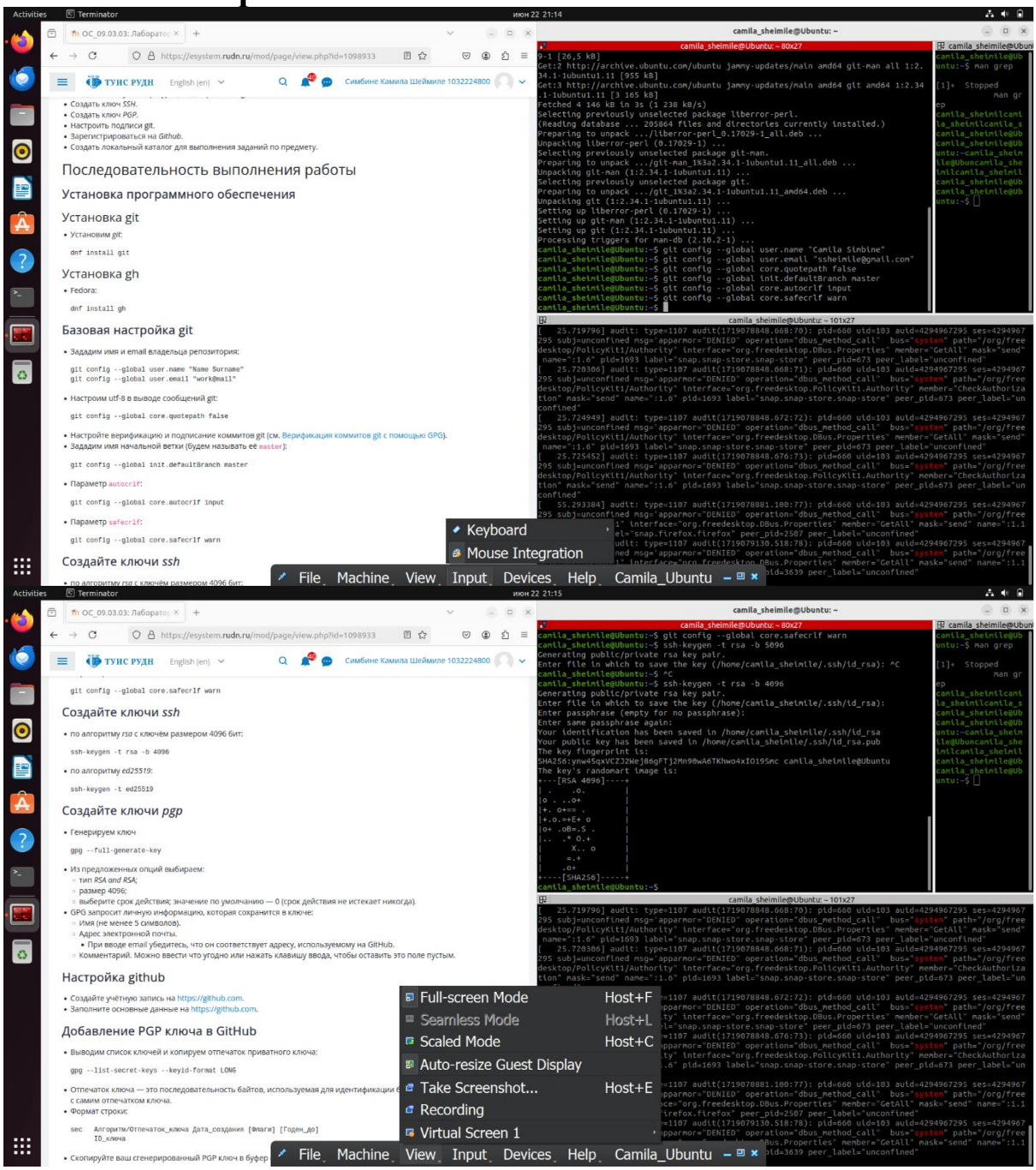
МОСКВА

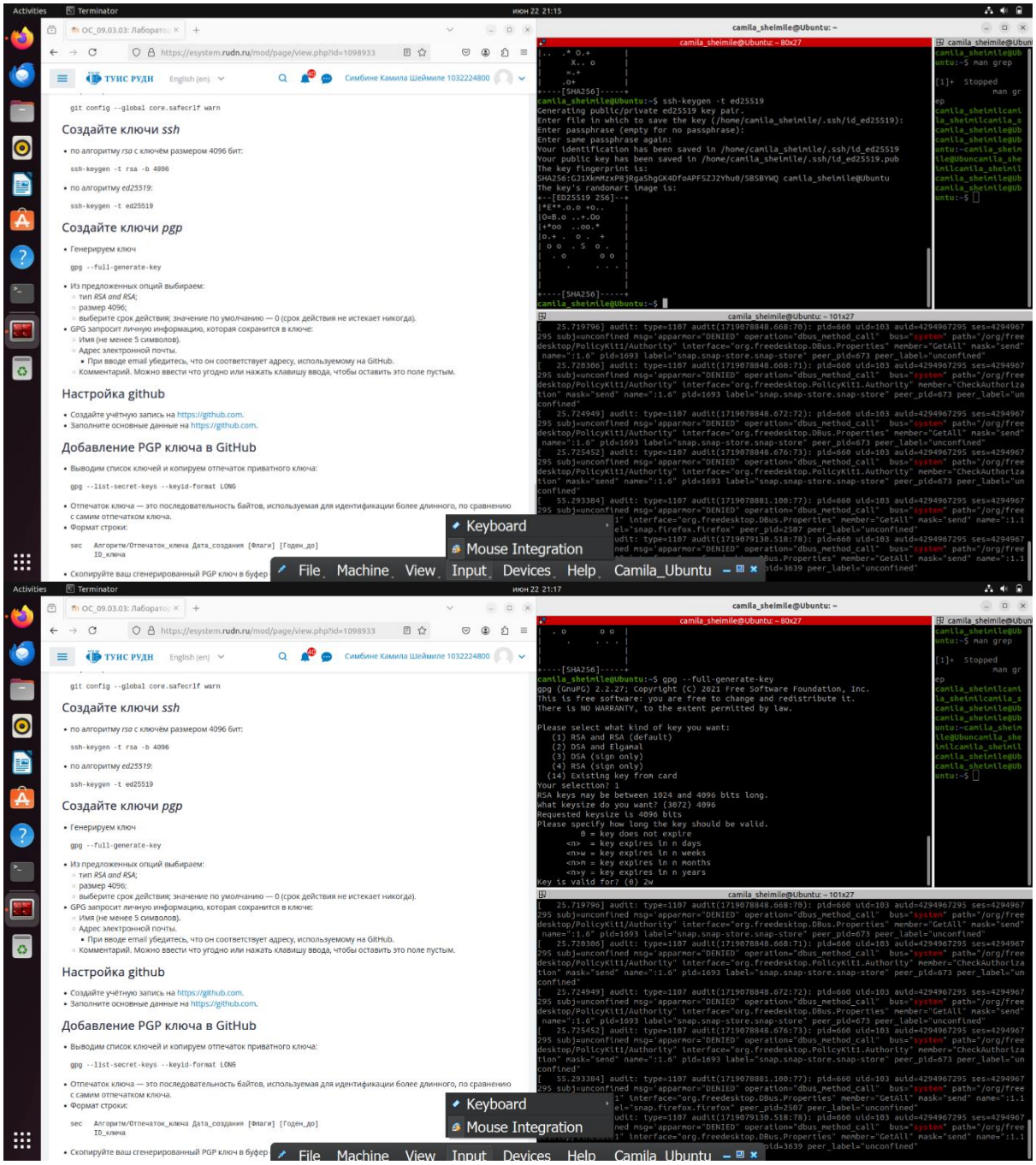
2024 г.

Цель работы

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

Ввыполнение работы






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

Создайте ключи *ssh*

- по алгоритму *rsa* с ключём размером 4096 бит:

```
ssh-keygen -t rsa -b 4096
```

- по алгоритму *ed25519*:

```
ssh-keygen -t ed25519
```

Создайте ключи *pgp*

- Генерируем ключ

```
gpg --full-generate-key
```

- Из предложенных опций выбираем:
  - тип *RSA and RSA*;
  - размер 4096;
  - выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).
- GPG запросит личную информацию, которая сохранится в ключе:
  - Имя (не менее 5 символов).
  - Адрес электронной почты.
    - При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.
  - Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

Настройка github

- Создайте учётную запись на https://github.com.
- Заполните основные данные на https://github.com.

Добавление PGP ключа в GitHub

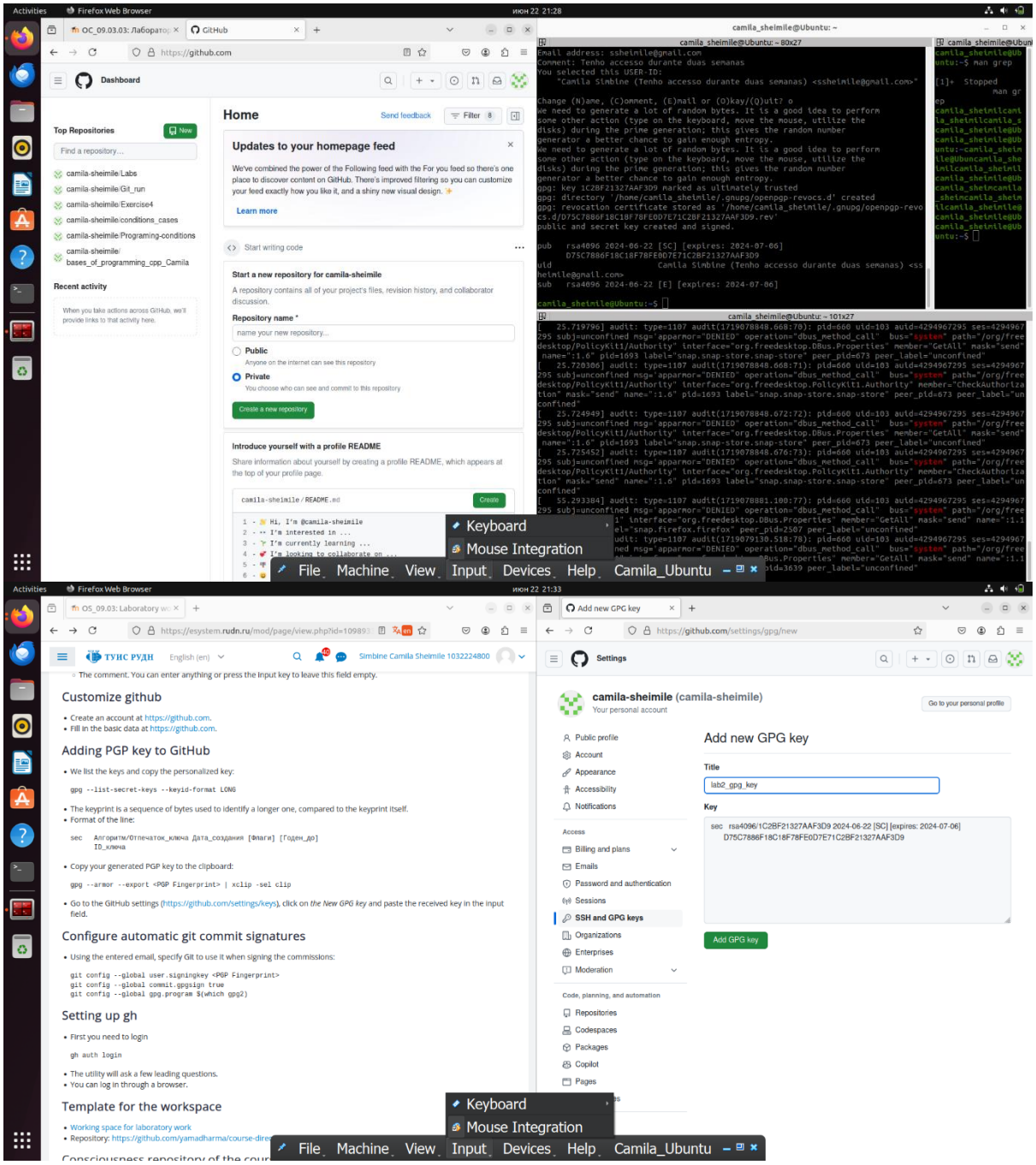
- Выводим список ключей и копируем отпечаток приватного ключа:

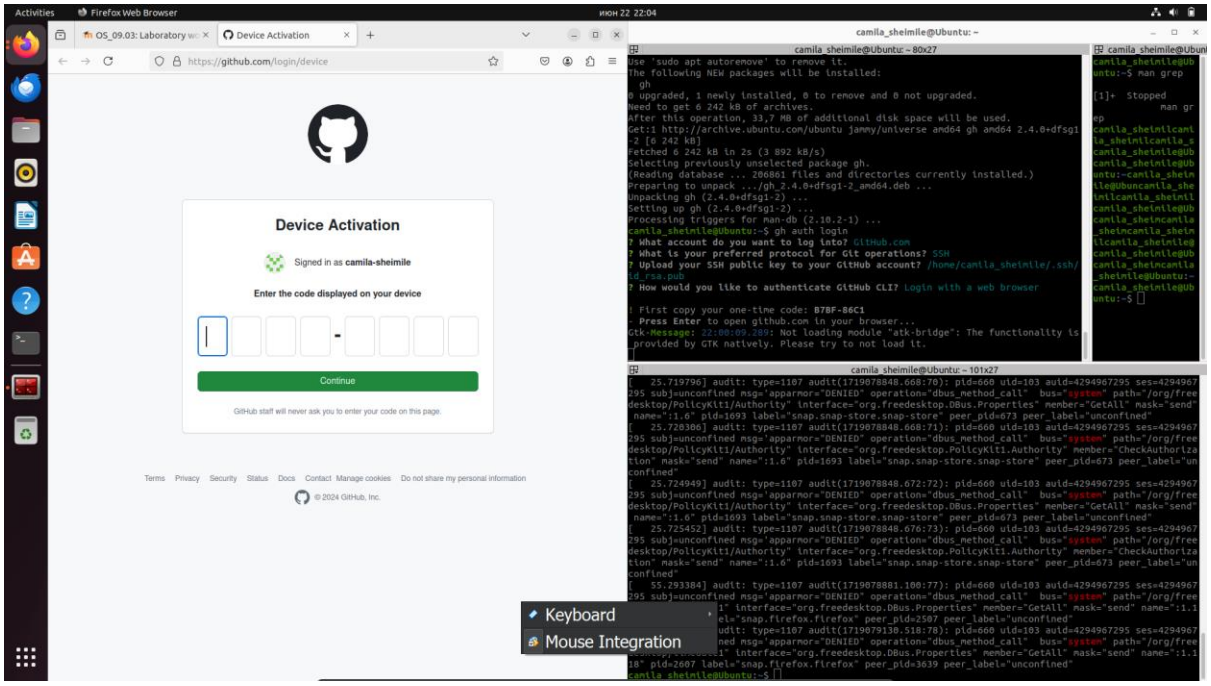
```
gpg --list-secret-keys --keyid-format LONG
```

- Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.
- Формат строки:

```
sec   Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до]
      ID_ключа
```

- Скопируйте ваш сгенерированный PGP ключ в буфер


Activities
Firefox Web Browser
июн 22 21:28
OC_09.03.03: Лаборатор
GitHub
https://github.com
Dashboard
Top Repositories
New
Find a repository...
camila-sheimile/Labs
camila-sheimile/Git_run
camila-sheimile/Exercise4
camila-sheimile/conditions_cases
camila-sheimile/Programing-conditions
camila-sheimile/bases_of_programming_cpp_Camila
Recent activity
When you take actions across GitHub, we'll provide links to that activity here.
Home
Send feedback
Filter
Updates to your homepage feed
We've combined the power of the Following feed with the For you feed so there's one place to discover content on GitHub. There's improved filtering so you can customize your feed exactly how you like it, and a shiny new visual design.
Learn more
Start writing code
Start a new repository for camila-sheimile
A repository contains all of your project's files, revision history, and collaborator discussion.
Repository name *
name your new repository...
Public
Anyone on the internet can see this repository
Private
You choose who can see and commit to this repository
Create a new repository
Introduce yourself with a profile README
Share information about yourself by creating a profile README, which appears at the top of your profile page.
camila-sheimile / README.md
Create
1 - Hi, I'm @camila-sheimile
2 - I'm interested in ...
3 - I'm currently learning ...
4 - I'm looking to collaborate on ...
5 -
6 -
camila_sheimile@Ubuntu: ~
camila_sheimile@Ubuntu: ~ 80x27
Email address: ssheimile@gmail.com
Comment: Tenho accesso durante duas semanas
You selected this USER-ID:
"Camila Simbine (Tenho accesso durante duas semanas) <ssheimile@gmail.com>"
Change (N)ame, (C)omment, (E)mail or (Q)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.
gpg: key 1C2BF21327AAF3D9 marked as ultimately trusted
gpg: directory '/home/camila_sheimile/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/camila_sheimile/.gnupg/openpgp-revocs.d/D75C7886F18C18F78FE0D7E71C2BF21327AAF3D9.rev'
public and secret key created and signed.
pub rsa4096 2024-06-22 [SC] [expires: 2024-07-06]
D75C7886F18C18F78FE0D7E71C2BF21327AAF3D9
uid Camila Simbine (Tenho accesso durante duas semanas) <ssheimile@gmail.com>
sub rsa4096 2024-06-22 [E] [expires: 2024-07-06]
camila_sheimile@Ubuntu:~$
camila_sheimile@Ubuntu: ~ 101x27
[25.719796] audit: type=1107 audit(1719078848.668:70): pid=660 uid=103 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/PolicyKit1/Authority" interface="org.freedesktop.DBus.Properties" member="GetAll" mask="send" name=":1.6" pid=1693 label="snap.snap-store.snap-store" peer_pid=673 peer_label="unconfined"
[25.720306] audit: type=1107 audit(1719078848.668:71): pid=660 uid=103 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/PolicyKit1/Authority" interface="org.freedesktop.PolicyKit1.Authority" member="CheckAuthorization" mask="send" name=":1.6" pid=1693 label="snap.snap-store.snap-store" peer_pid=673 peer_label="unconfined"
[25.724949] audit: type=1107 audit(1719078848.672:72): pid=660 uid=103 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/PolicyKit1/Authority" interface="org.freedesktop.DBus.Properties" member="GetAll" mask="send" name=":1.6" pid=1693 label="snap.snap-store.snap-store" peer_pid=673 peer_label="unconfined"
[25.725452] audit: type=1107 audit(1719078848.676:73): pid=660 uid=103 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/PolicyKit1/Authority" interface="org.freedesktop.PolicyKit1.Authority" member="CheckAuthorization" mask="send" name=":1.6" pid=1693 label="snap.snap-store.snap-store" peer_pid=673 peer_label="unconfined"
[55.293384] audit: type=1107 audit(1719078881.180:77): pid=660 uid=103 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/PolicyKit1/Authority" interface="org.freedesktop.DBus.Properties" member="GetAll" mask="send" name=":1.1
el="snap.firefox.firefox" peer_pid=2507 peer_label="unconfined"
udit: type=1107 audit(1719079130.518:78): pid=660 uid=103 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/DBus.Properties" member="GetAll" mask="send" name=":1.1
pid=3639 peer_label="unconfined"
Keyboard
Mouse Integration
File Machine View Input Devices Help Camila_Ubuntu
Activities
Firefox Web Browser
июн 22 21:33
OS_09.03: Laboratory wo
https://esystem.rudn.ru/mod/page/view.php?id=109893
ТУИС РУДН
English (en)
Simbine Camila Sheimile 1032224800
The comment. You can enter anything or press the Input key to leave this field empty.
Customize github
Create an account at https://github.com.
Fill in the basic data at https://github.com.
Adding PGP key to GitHub
We list the keys and copy the personalized key:
gpg --list-secret-keys --keyid-format LONG
The keyprint is a sequence of bytes used to identify a longer one, compared to the keyprint itself.
Format of the line:
sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до]
ID_ключа
Copy your generated PGP key to the clipboard:
gpg --armor --export <PGP Fingerprint> | xclip -sel clip
Go to the GitHub settings (https://github.com/settings/keys), click on the New GPG key and paste the received key in the input field.
Configure automatic git commit signatures
Using the entered email, specify Git to use it when signing the commissions:
git config --global user.signingkey <PGP Fingerprint>
git config --global commit.gpgsign true
git config --global gpg.program $(which gpg2)
Setting up gh
First you need to login
gh auth login
The utility will ask a few leading questions.
You can log in through a browser.
Template for the workspace
Working space for laboratory work
Repository: https://github.com/yamadharma/course-dire
Consciousness repository of the cour
Add new GPG key
https://github.com/settings/gpg/new
Settings
camila-sheimile (camila-sheimile)
Your personal account
Go to your personal profile
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Code, planning, and automation
Repositories
Codespaces
Packages
Copilot
Pages
Add new GPG key
Title
lab2_gpg_key
Key
sec rsa4096/1C2BF21327AAF3D9 2024-06-22 [SC] [expires: 2024-07-06]
D75C7886F18C18F78FE0D7E71C2BF21327AAF3D9
Add GPG key
Keyboard
Mouse Integration
File Machine View Input Devices Help Camila_Ubuntu



Контрольные вопросы

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

Системы контроля версий (Version Control Systems, VCS) — это инструменты, предназначенные для управления изменениями в файлах и координации работы над этими файлами среди нескольких человек. Они позволяют:

- Отслеживать изменения в коде и других файлах.
- Восстанавливать предыдущие версии файлов.
- Работать в команде, сливая изменения от разных участников.
- Создавать ветки для независимой разработки различных функций или исправлений.

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

- Хранилище (repository): Центральное место, где хранится весь код и история изменений. Может быть локальным (на компьютере разработчика) или удалённым (на сервере).
- Commit: Операция, фиксирующая изменения в хранилище. Каждый commit представляет собой снимок состояния файлов на данный момент.

- История (history): Последовательность commit'ов, показывающая все изменения, сделанные в проекте с течением времени.
- Рабочая копия (working copy): Локальная копия файлов, над которой работает разработчик. Изменения в этой копии могут быть зафиксированы (committed) в хранилище.

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

- Централизованные VCS: Имеют один центральный сервер, который содержит все данные и историю изменений. Пользователи получают рабочие копии файлов, но все операции фиксируются на сервере. Примеры: Subversion (SVN), Perforce.
- Децентрализованные (распределённые) VCS: Каждый пользователь имеет полную копию хранилища, включая всю историю изменений. Это позволяет работать автономно и синхронизировать изменения с другими пользователями. Примеры: Git, Mercurial.

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

При единоличной работе с хранилищем действия включают:
1. Инициализация хранилища (если оно ещё не создано):
   git init
2. Добавление файлов в хранилище:
   git add <файл>
3. Фиксация изменений:
   git commit -m "Описание изменений"
4. Просмотр истории изменений:
   git log

5. Опишите порядок работы с общим хранилищем VCS.

1. Клонирование удалённого репозитория:
   git clone <URL>
2. Создание новой ветки (при необходимости):

git checkout -b <branch-name>
3. Работа с файлами (изменение, добавление, удаление).
4. Фиксация изменений:
git add <файл>
git commit -m "Описание изменений"
5. Обновление локальной копии из удалённого репозитория:
git pull
6. Отправка изменений в удалённый репозиторий:
git push

6. Каковы основные задачи, решаемые инструментальным средством git?

Git решает следующие основные задачи:
- Управление версиями кода и документов.
- Совместная работа над проектами.
- Поддержка ветвления и слияния.
- Хранение полной истории изменений.
- Поддержка работы в распределённой среде.

7. Назовите и дайте краткую характеристику командам git.

- git init: Инициализация нового локального репозитория.
- git clone <URL>: Клонирование удалённого репозитория.
- git add <файл>: Добавление файла в индекс для последующего коммита.
- git commit -m "сообщение": Фиксация изменений в репозитории с сообщением.
- git status: Отображение состояния рабочего каталога и индекса.
- git log: Просмотр истории коммитов.
- git pull: Получение изменений из удалённого репозитория и слияние с локальной веткой.
- git push: Отправка изменений в удалённый репозиторий.
- git branch: Список веток или создание новой ветки.
- git checkout <ветка>: Переключение на другую ветку.
- git merge <ветка>: Слияние указанной ветки с текущей.

8. Примеры использования при работе с локальным и удалённым репозиториями.

Локальный репозиторий:
Инициализация нового репозитория
git init

Добавление файла и коммит
echo "Hello, World!" > hello.txt
git add hello.txt
git commit -m "Add hello.txt"

Удалённый репозиторий:
Клонирование удалённого репозитория
git clone https://github.com/user/repo.git

Изменение файла и отправка изменений
cd repo
echo "New Line" >> hello.txt
git add hello.txt
git commit -m "Update hello.txt"
git push

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

Ветви (branches) в Git позволяют параллельно работать над разными задачами, функциями или исправлениями. Они помогают:
- Разделять рабочие процессы (например, разработка новых функций и исправление багов).
- Изолировать изменения до тех пор, пока они не будут готовы к интеграции.
- Проводить эксперименты без риска повредить основную ветку кода.

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

Иногда необходимо игнорировать определённые файлы (например, временные файлы, конфигурационные файлы, артефакты сборки). Для этого используется файл `.gitignore`, в который добавляются шаблоны для игнорируемых файлов и директорий. Это позволяет избежать случайного добавления ненужных файлов в репозиторий и поддерживать его чистоту.

Пример файла `.gitignore`:
Игнорирование всех файлов .log
*.log

Игнорирование директории build/
build/

Игнорирование файла конфигурации
config.yml

Заключение

Эти контрольные вопросы охватывают основные концепции и команды, связанные с системами контроля версий, такими как Git. Понимание этих концепций важно для эффективной работы с VCS как в одиночку, так и в команде.