# 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

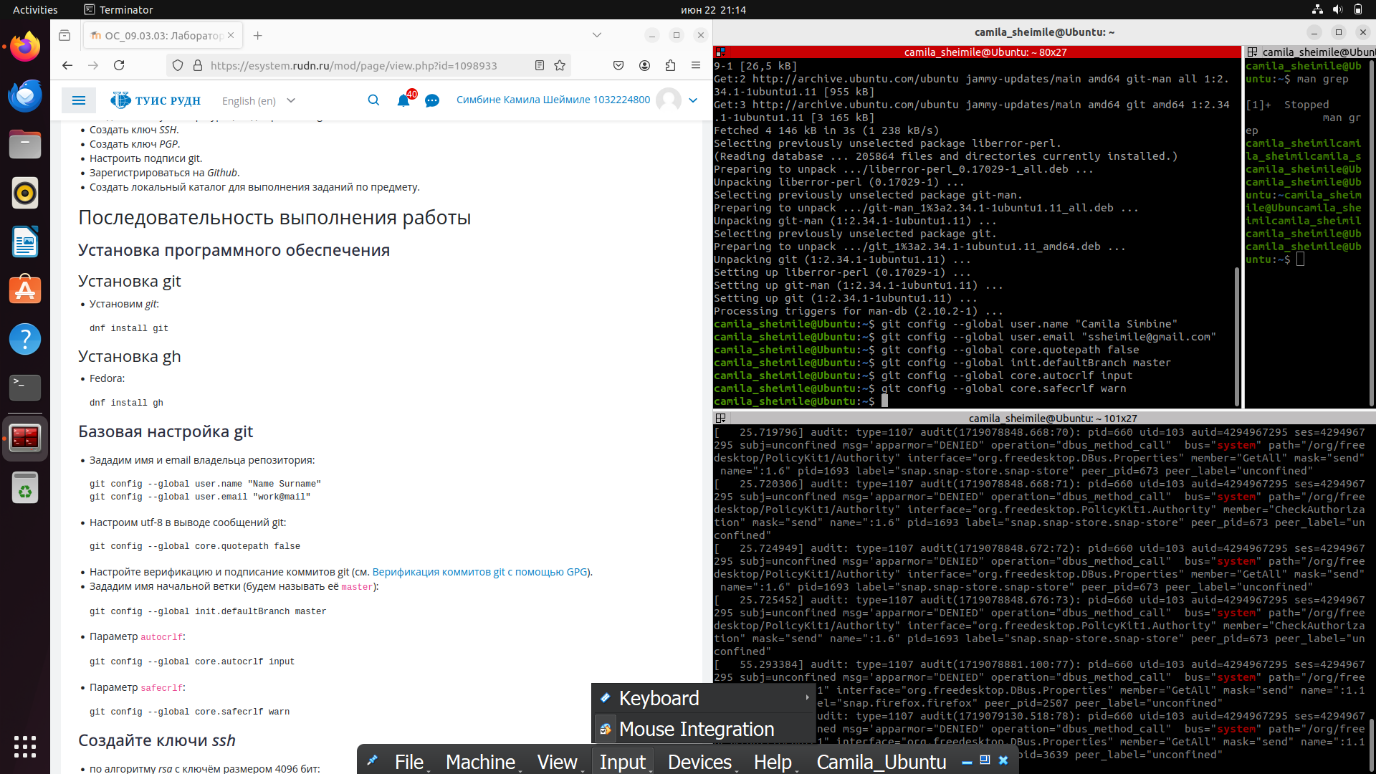
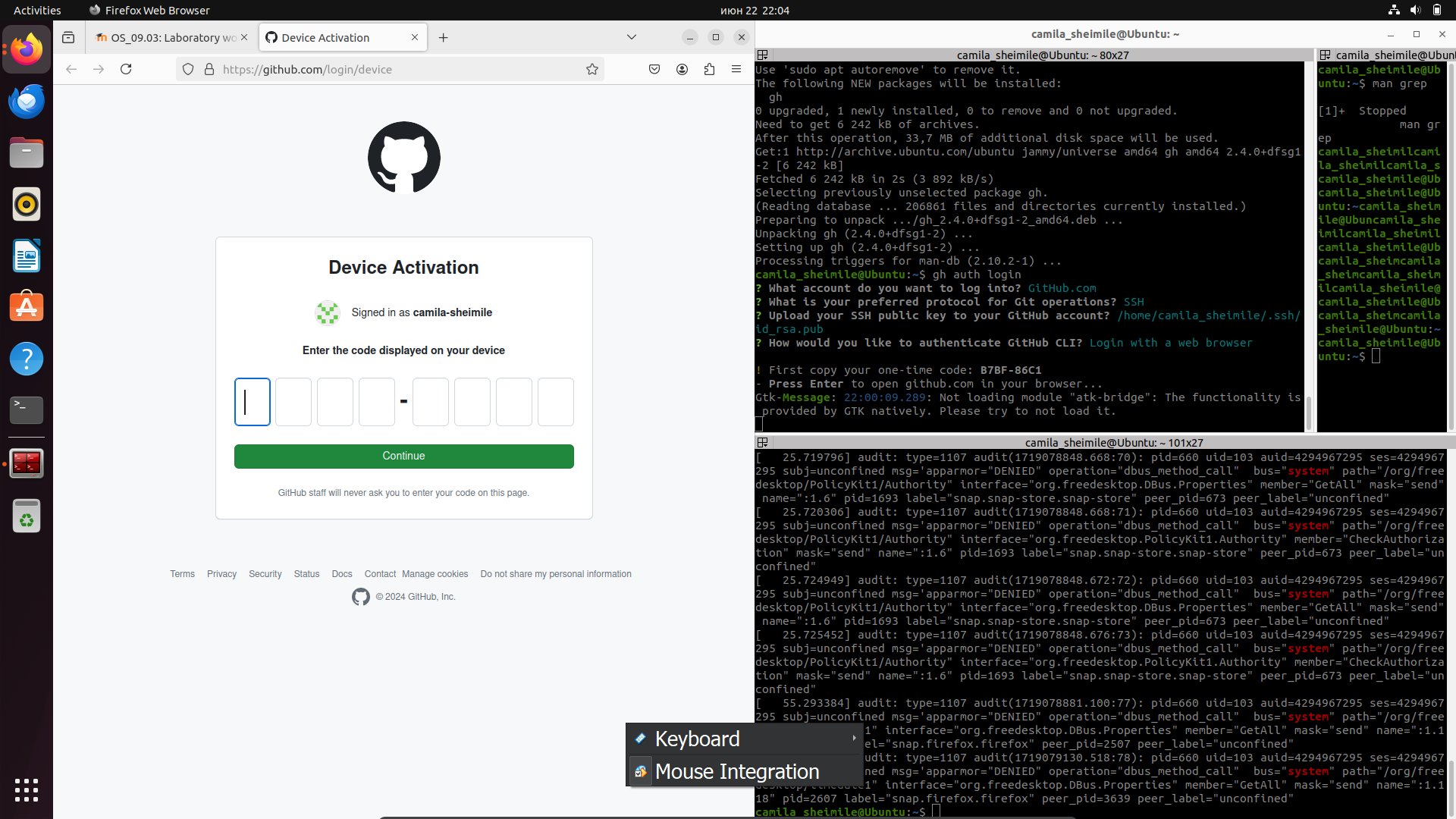
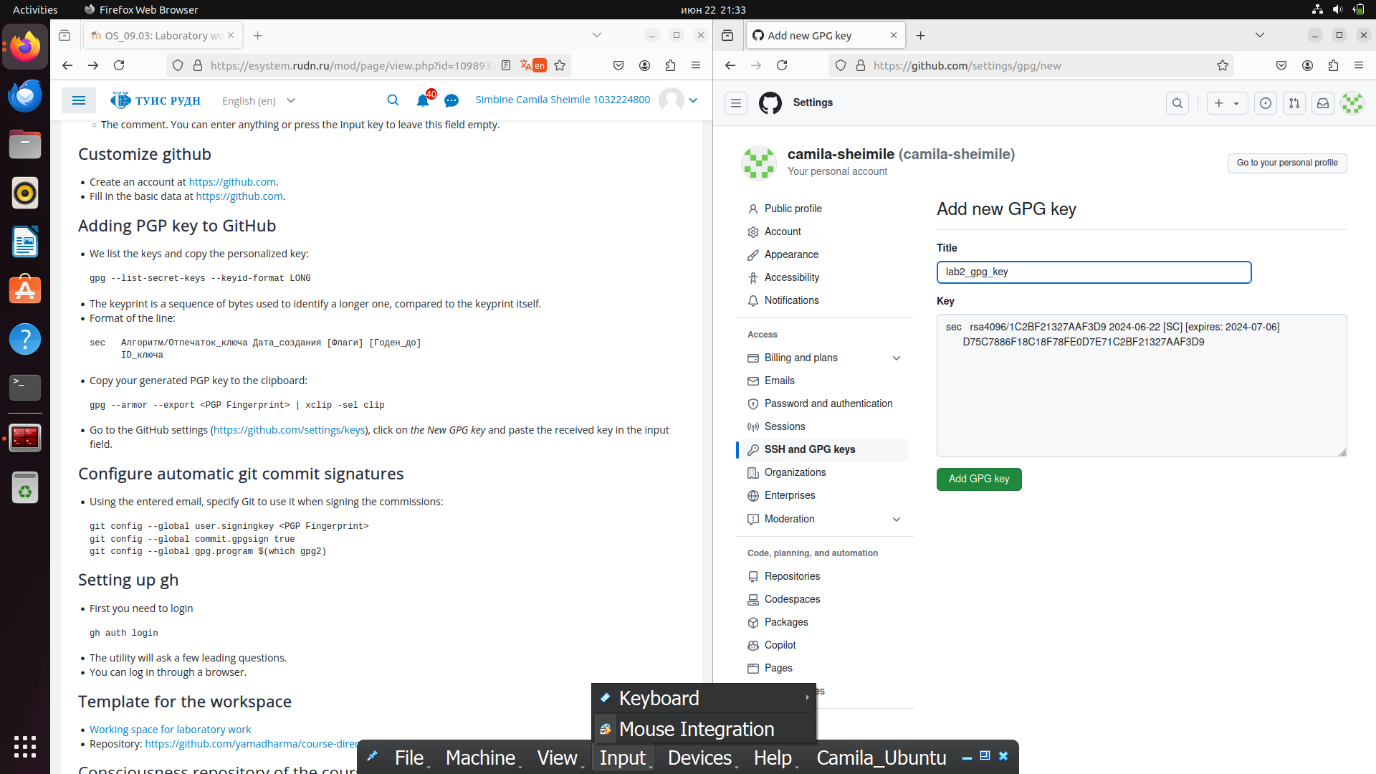
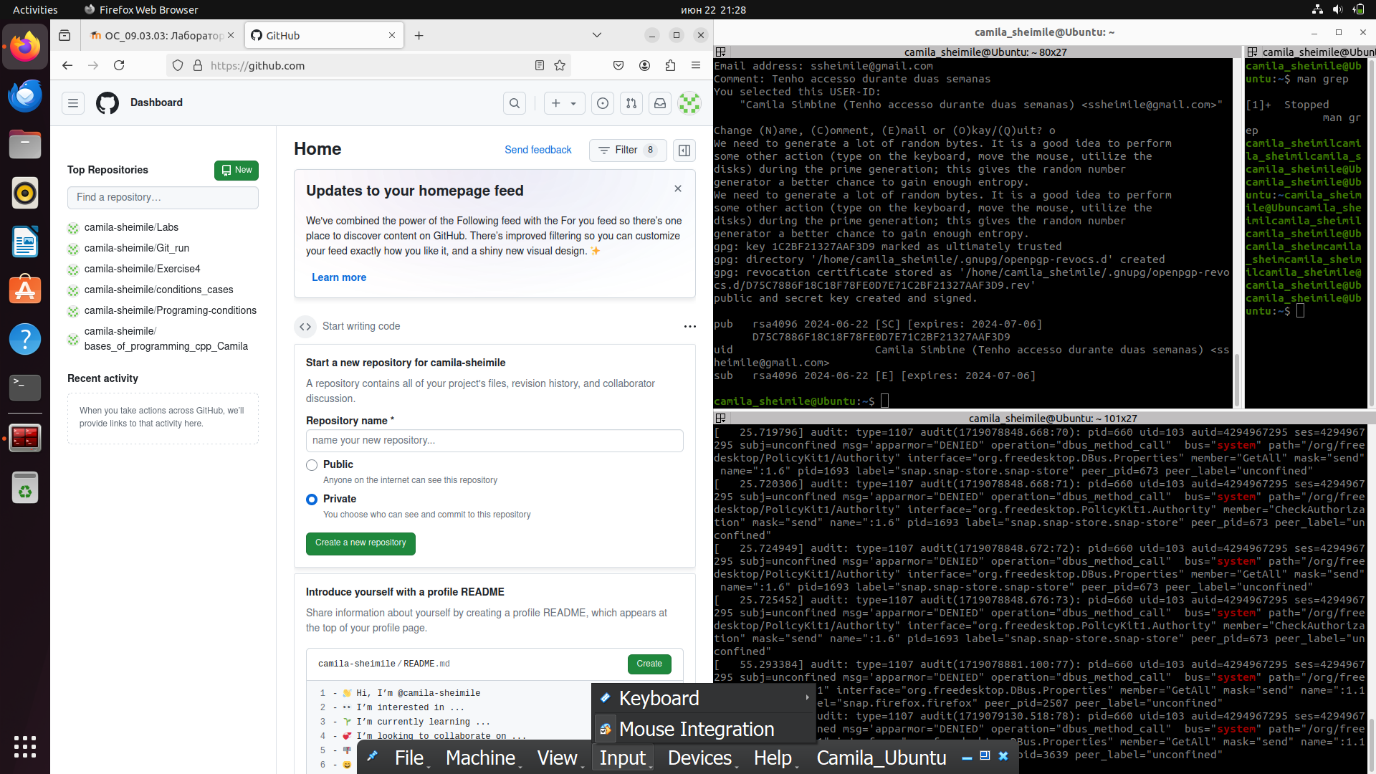
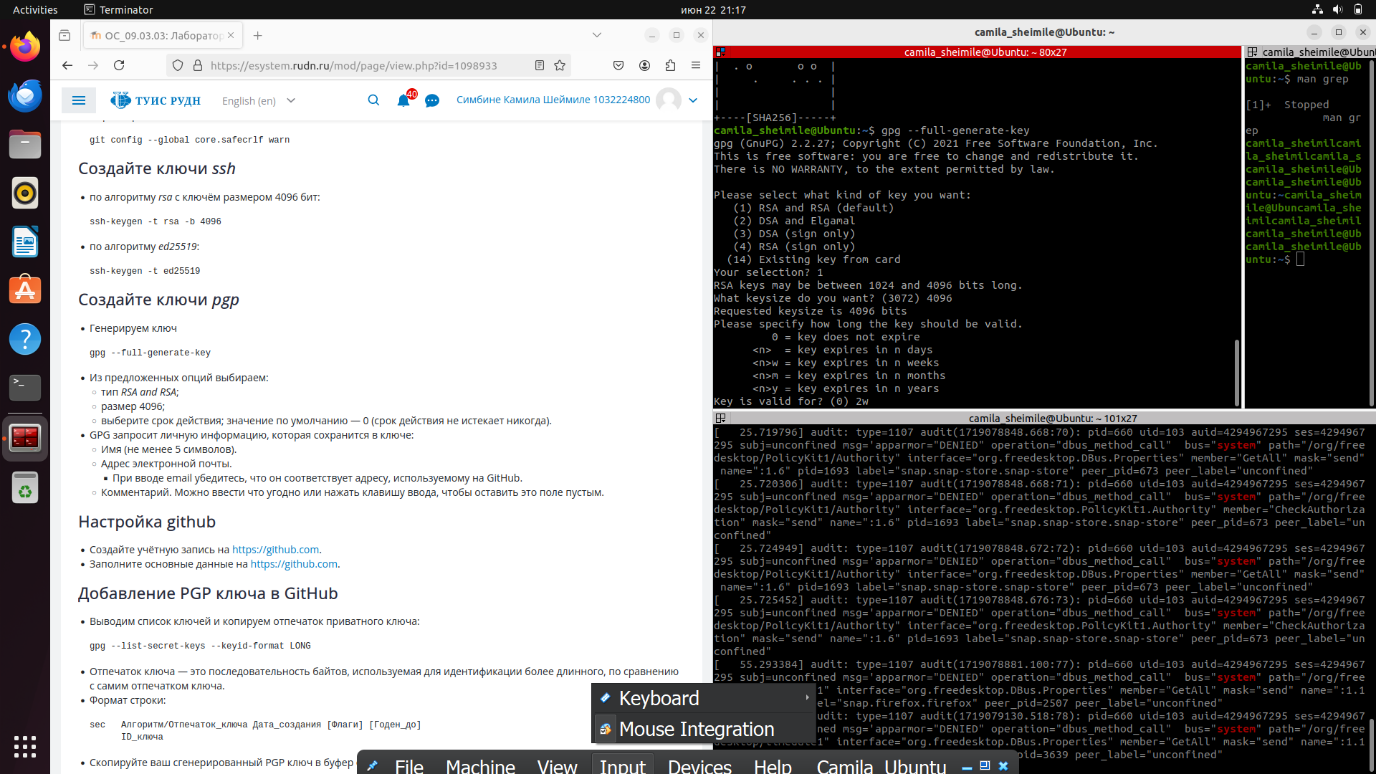
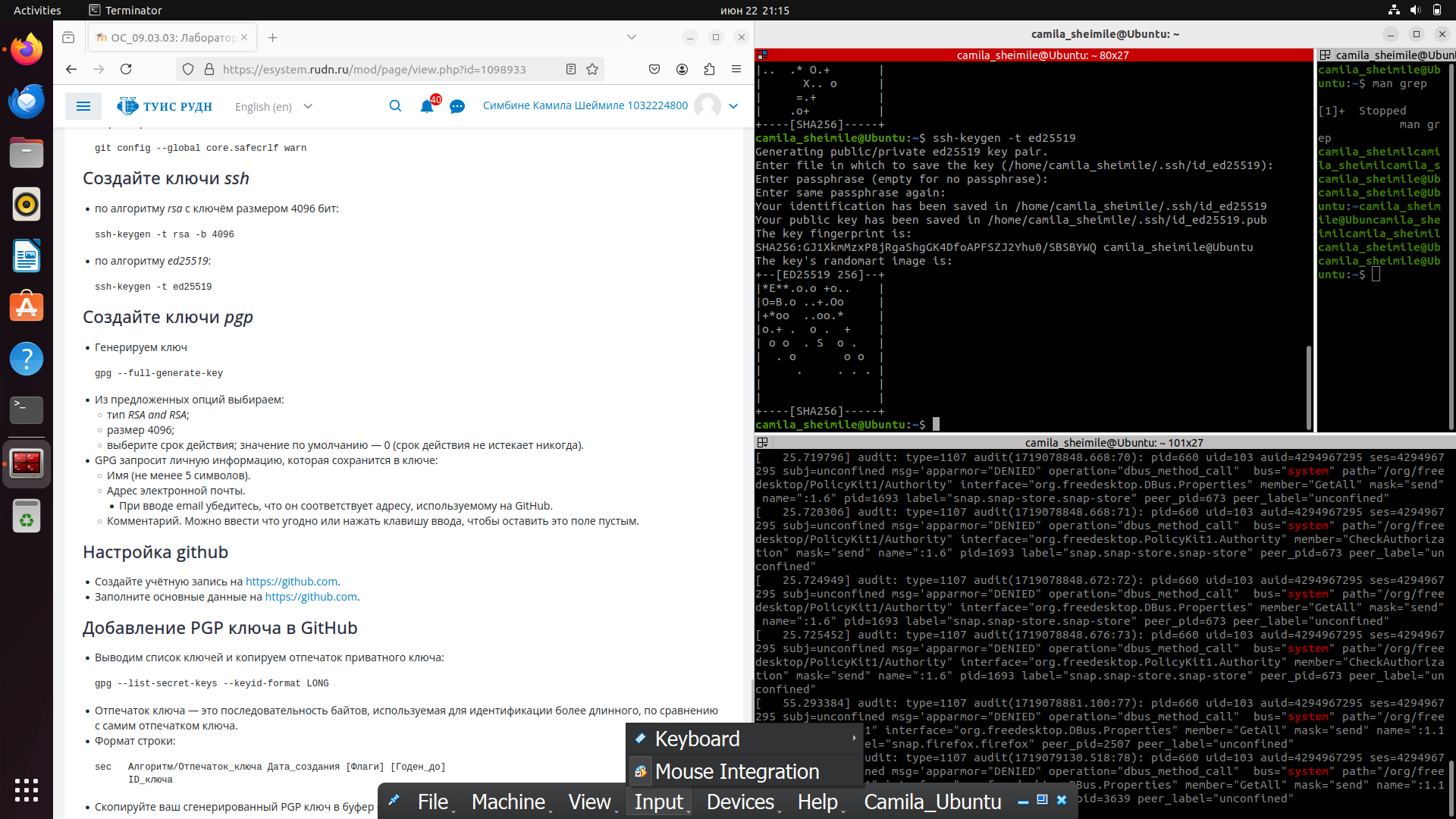
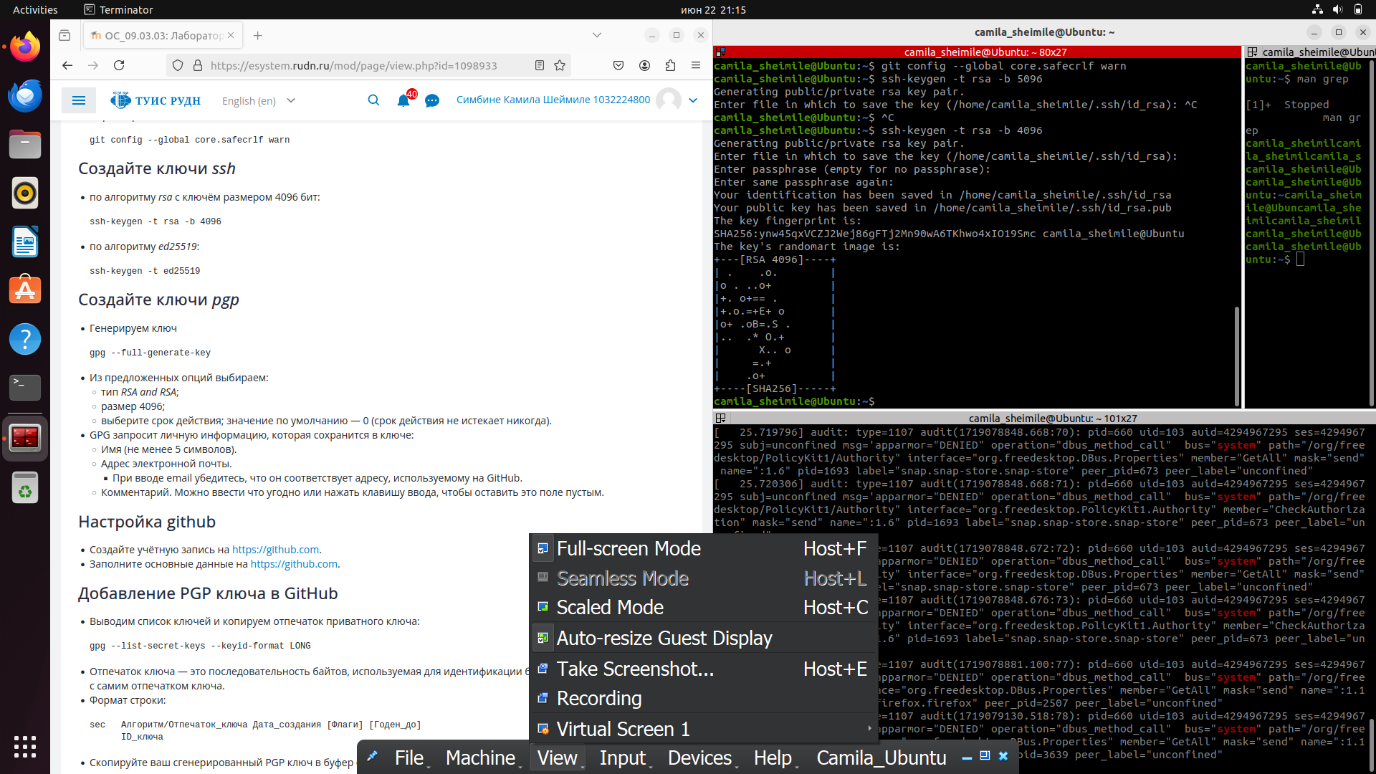
Группа: НПИбд-03-23

**МОСКВА**

2024 г.

**Цель работы**

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

**Ввыполнение работы******

**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.