

# **Отчёт по лабораторной работе №2**

**Дисциплина: Архитектура компьютера и операционные системы**

Ванюшкина Татьяна Валерьевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>4</b>	<b>Выводы</b>	<b>12</b>
	<b>Список литературы</b>	<b>13</b>

## **Список иллюстраций**

## **Список таблиц**

# 1 Цель работы

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

## 2 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.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию,

отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

## 3 Выполнение лабораторной работы

### 1. Установка программного обеспечения

Устанавливаем git

```
[liveuser@localhost-live ~]$ sudo -i
[root@localhost-live ~]# dnf install git
Updating and loading repositories:
Fedora 41 openh264 (From Cisco) - x86_64
Fedora 41 - x86_64 - Updates
Fedora 41 - x86_64
Repositories loaded.
```

(рис.1 ??) {#fig:001}

Устанавливаем gh

```
[root@localhost-live ~]# dnf install gh
Updating and loading repositories:
Repositories loaded.
```

(рис.2 ??) {#fig:002}

### 2. Базовая настройка git

Задаем имя и email владельца репозитория

```
complete
[root@localhost-live ~]# git config --global user.name "Tatyana Vanyushkina"
[root@localhost-live ~]# git config --global user.email "Vanyushkina.t@list.ru"
```

(рис.3 ??) {#fig:003}

Задаем имя начальной ветки

```
[root@localhost-live ~]# git config --global init.defaultBranch master
```

(рис.4 ??) {#fig:004}

Параметр autocrlf

```
[root@localhost-live ~]# git config --global core.autocrlf input
```

(рис.5 ??) {#fig:005}

Параметр safecrlf

```
[root@localhost-live ~]# git config --global core.safecrlf warn
```

(рис.6 ??) {#fig:006}

### 3. Создание ключей



Создаем ключи

```
[root@localhost-live ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): ssh-keygen -t ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh-keygen -t ed25519
Your public key has been saved in ssh-keygen -t ed25519.pub
The key fingerprint is:
SHA256:JET799eIXgwL20LiGAARvurpps1DuTFbi3rIEwi9S4 root@localhost-live
```

(рис.7 ??) {#fig:007}

Генерируем ключ

```
[root@localhost-live ~]# gpg --full-generate-key
```

(рис.9 ??) {#fig:009}

Выводим список ключей, копируем отпечаток приватного ключа и копируем  
генерированный PGP ключ в буфер обмена

```
[root@localhost-live ~]# gpg --list-secret-keys --keyid-format LONG
gpg: /root/.gnupg/trustdb.gpg: trustdb created
[root@localhost-live ~]# gpg --armor --export <PGP Fingerprint> | xclip -sel clip
```

(рис.10 ??) {#fig:010}

Перехожу в настройки GitHub (<https://github.com/settings/keys>), нажимаю на кнопку New GPG key и вставляю полученный ключ в поле ввода.

## Deploy keys / Add new

Title

my key

Key

EF66AF49608ED501

(рис.11 ??) {#fig:011}

### 3. Настройка gh

Авторизируемся

```
[root@localhost-live Операционные системы]# gh auth
login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 16D7-300B
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as tatyana952
```

(рис.12 ??) {#fig:012}

Создаем репозиторий

```
[root@localhost-live ~]# mkdir -p ~/work/study/2024-2025/"
Операционные системы"
[root@localhost-live ~]# cd ~/work/study/2024-2025/"
Операционные системы"
[root@localhost-live ^JОперационные системы]# gh repo create study_2024-2025_o
s-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository tatyana952/study_2024-2025_os-intro on GitHub
https://github.com/tatyana952/study_2024-2025_os-intro
[root@localhost-live ^JОперационные системы]# git clone --recursive git@github
b.com:<owner>/study_2024-2025_os-intro.git os-intro
```

(рис.13 ??) {#fig:013}

Удаляем лишние файлы

```
[root@localhost-live os-intro]# rm package.json
```

(рис.14 ??) {#fig:014}

Создаем необходимые каталоги и добавляем файлы на гитхаб

```
[root@localhost-live os-intro]# echo os-intro > COURSE
```

(рис.15 ??) {#fig:015}















tatyana952 /  
study\_2024-2025\_os-intro

(рис.16 ??) {#fig:016}

```
[root@localhost-live os-intro]# echo os-intro > COURSE
```

(рис.15 ??) {#fig:015}

 <b>tatyana952</b> Initial commit		912d1ca · 22 minutes ago 
 config	Initial commit	22 minutes ago
 template	Initial commit	22 minutes ago
 .gitattributes	Initial commit	22 minutes ago
 .gitignore	Initial commit	22 minutes ago
 .gitmodules	Initial commit	22 minutes ago
 CHANGELOG.md	Initial commit	22 minutes ago
 COURSE	Initial commit	22 minutes ago
 LICENSE	Initial commit	22 minutes ago
 Makefile	Initial commit	22 minutes ago
 README.en.md	Initial commit	22 minutes ago
<a href="#">View all files</a>		

(рис.17 ??)

{#fig:017}

## 4 Выводы

я изучила идеологию и применение средств контроля версий и освоила умения по работе с git

# Список литературы

Курс: Архитектура компьютеров и операционные системы. Раздел “Операционные системы” (02.03.00, УГСН) (rudn.ru)