

Отчет по лабораторной работе №2

Операционные системы

Монхжаргал Тувшинбаяр

Содержание

Список иллюстраций

3.1	Установка git и gh	7
•	Задаю имя и email владельца репозитория	7
•	Настройка utf-8 в выводе сообщений git	8
3.4	Задаю имя начальной ветки	8
•	Задаю параметры autocrlf и safecrlf	8
•	Генерация ssh ключа по алгоритму rsa	8
•	Генерация ssh ключа по алгоритму ed25519	9
•	Генерация ключа.....	10
•	Защита ключа GPG	10
•	Аккаунт на Github.....	11
•	Вывод списка ключей	11
•	Копирование ключа в буфер обмена.....	12
•	Настройки GitHub	12
•	Добавление нового PGP ключа	12
•	Добавленный ключ GPG.....	13

• Настройка подписей Git.....	13
• Авторизация в gh	13
• Завершение авторизации через браузер	14
• Завершение авторизации	14
• Создание репозитория.....	15
• Перемещение между директориями	15
• Удаление файлов и создание каталогов.....	15
• Отправка файлов на сервер	15
• Отправка файлов на сервер	16

Список таблиц

• Цель работы

Цель данной лабораторной работы – изучение идеологии и применения средств контроля версий, освоение умения по работе с git.

• Задание

- Создать базовую конфигурацию для работы с git
- Создать ключ SSH

- Создать ключ GPG
- Настроить подписи Git
- Зарегистрироваться на GitHub
- Создать локальный каталог для выполнения заданий по предмету.

• Выполнение лабораторной работы

• Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: `dnf install git` и `dnf install gh` (рис. 3.1).

```
liveuser@tmunkhjargal:~$ sudo dnf -y install git
Fedora 40 - x86_64                               2.7 MB/s | 20 MB    00:07
Fedora 40 openh264 (From Cisco) - x86_64        1.2 kB/s | 1.4 kB   00:01
Fedora 40 - x86_64 - Updates                    5.5 MB/s | 9.8 MB   00:01
Package git-2.44.0-1.fc40.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
```

Рис. 3.1: Установка git и gh

• Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис. 3.2).

```
liveuser@tmunkhjargal:~$ git config --global user.name "mtuvshinbayar"
liveuser@tmunkhjargal:~$ git config --global user.email "1032235566@pfur.ru"
```

Рис. 3.2: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного отображения (рис. 3.3).

```
liveuser@tmunkhjargal:~$ git config --global core.quotePath false  
liveuser@tmunkhjargal:~$ git config --global init.defaultBranch master
```

Рис. 3.3: Настройка utf-8 в выводе сообщений git Начальной ветке

задаю имя master (рис. 3.4).

Рис. 3.4: Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf для корректного отображения конца строки(рис. 3.5).

```
liveuser@tmunkhjargal:~$ git config --global core.autocrlf input  
liveuser@tmunkhjargal:~$ git config --global core.safecrlf warn
```

Рис. 3.5: Задаю параметры autocrlf и safecrlf

- **Создание ключа SSH**

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис. 3.6).

```

liveuser@tmunkhjargal:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/liveuser/.ssh/id_rsa):
Created directory '/home/liveuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/liveuser/.ssh/id_rsa
Your public key has been saved in /home/liveuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ECpAP9J2UwTeHLStTVBc7vjrE+YqN8U+TFC+rIsaHSI liveuser@tmunkhjargal
The key's randomart image is:
+---[RSA 4096]-----+
|o.  .=*o...|
| .o  ..+.=...|
| ..=..+ oo. |
| o.o  ..+.o. |
| E . oSo+.. |
| . o . .0 |
| . . 0.. |
| ...+ *. |
| ...oo=oo |
+-----[SHA256]-----+

```

Рис. 3.6: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. 3.7).

```

liveuser@tmunkhjargal:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/liveuser/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/liveuser/.ssh/id_ed25519
Your public key has been saved in /home/liveuser/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ZBx7qV3HSYbKYIMpjpniOB6Hy8CZj8GfiGgQ74N4jLU liveuser@tmunkhjargal
The key's randomart image is:
+--[ED25519 256]--+
|      o.  .o |
|o  . o.+o ..+ .|
|*++= . .+o.. + |
|=X*.. o +o. . |
|O*X.. S . |
|*BE+ |
|.. . |
| |
+-----[SHA256]-----+

```

Рис. 3.7: Генерация ssh ключа по алгоритму ed25519

• Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максим- мальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. 3.8).

```

liveuser@tmunkhjargal:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 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.

gpg: directory '/home/liveuser/.gnupg' created
Please select what kind of key you want:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (sign only)
 (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all

```

Рис. 3.8: Генерация ключа

Ввожу фразу-пароль для защиты нового ключа (рис. 3.9).

(Забла

сфотографировать)Рис.

3.9: Защита ключа GPG

• Регистрация на Github

У меня уже был создан аккаунт на Github, соответственно, основные данные аккаунта я так же заполняла и проводила его настройку, поэтому просто вхожу в свой аккаунт (рис. 3.10).



mtuvshinbayar

Set status

Edit profile

Popular repositories

[Customize your pins](#)

[study_2023-2024_arch-pc](#)

Public

Shell

[study_2023-2024_arh-pc](#)

Public

Рис. 3.10: Аккаунт на Github

• Добавление ключа GPG в Github

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа (последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком, ключа), он стоит после знака слеша, копирую его в буфер обмена (рис. 3.11).

```
liveuser@tmunkhjargal:~$ gpg --list-secret-key --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
[keyboard]
-----
sec   rsa4096/3029A538EE593CB5 2024-09-04 [SC]
      45157BEF8145E8591D8D3FA63029A538EE593CB5
uid           [ultimate] mtuvshinbayar <1032235566@pfur.ru>
ssb   rsa4096/29D6D35142ED77F7 2024-09-04 [E]
```

Рис. 3.11: Вывод списка ключей

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает утилита xclip (рис. 3.12).

```
liveuser@tmunkhjargal:~$ gpg --armor --export 3029A538EE593CB5 | xclip -sel clip
```

Рис. 3.12: Копирование ключа в буфер обмена

Открываю настройки GitHub, ищу среди них добавление GPG ключа (рис. 3.13).

GPG keys

New GPG key

There are no GPG keys associated with your account.

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

Рис. 3.13: Настройки GitHub

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис. 3.14).

Add new GPG key

Title

Key

-----BEGIN PGP PUBLIC KEY BLOCK-----

```
mQINBGbYwScBEADFFSdJL5TUH8TNnYrIVsXU26VNgTtZy3QPca/cLTUSFF8EhhZa
HgKswbVZUNzYH3ss5RokFao3cMdsVgvSDdPgLXXFDNZJZvAJ6vYWm+BkeqflgEsT
8r9+NRDkmyDsmz+k5e9IVL7zLTLWRSXXORwwrRIYcnPVWkkFXXy8gPDBDtQ5gy9
Mjmh/i6YIEo0mhXAY5Ai6MJT51FZwiksKlJB2MrUPnLFTNSIWSVfXejMGYR//XVvm
fjY2wZPwt1J56vxDX0mnaH5Rhb/3TUKTcEjQp3kNA1RULIab0fOWjQYqfkblysSu
u7IBWB70w82Jcq6+jR1xn6BKdk874MEkSGMLViW18MAJH5z6fYAr91Mnj8nVVBv
dALCvYFGvt70iFVV5WFIV2ixnBnZWYfWnHGWLtQwWciGveQJ9nLQmpUCU7JNnut
```


Add GPG key

Рис. 3.14: Добавление нового PGP ключа

Я добавила ключ GPG на GitHub (рис. 3.15).

PGP keys

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


GPG

Email address: 1032235566@pfur.ru

Key ID: 3029A538EE593CB5

Subkeys: 29D6D35142ED77F7

Added on Sep 4, 2024

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

Рис. 3.15: Добавленный ключ GPG

- **Настроить подписи Git**

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов(рис. 3.16).

```
liveuser@tmunkhjargal:~$ git config --global user.signingkey 3029A538EE593CB5
liveuser@tmunkhjargal:~$ git config --global commit.gpgsign true
liveuser@tmunkhjargal:~$ git config --global gpg.program $(which gpg2)
```

Рис. 3.16: Настройка подписей Git

- **Настройка gh**

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис. 3.17).

```
liveuser@tmunkhjargal:~$ gh auth login
? What account do you want to log into? 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
```

Рис. 3.17: Авторизация в gh

Завершаю авторизацию на сайте (рис. 3.18).

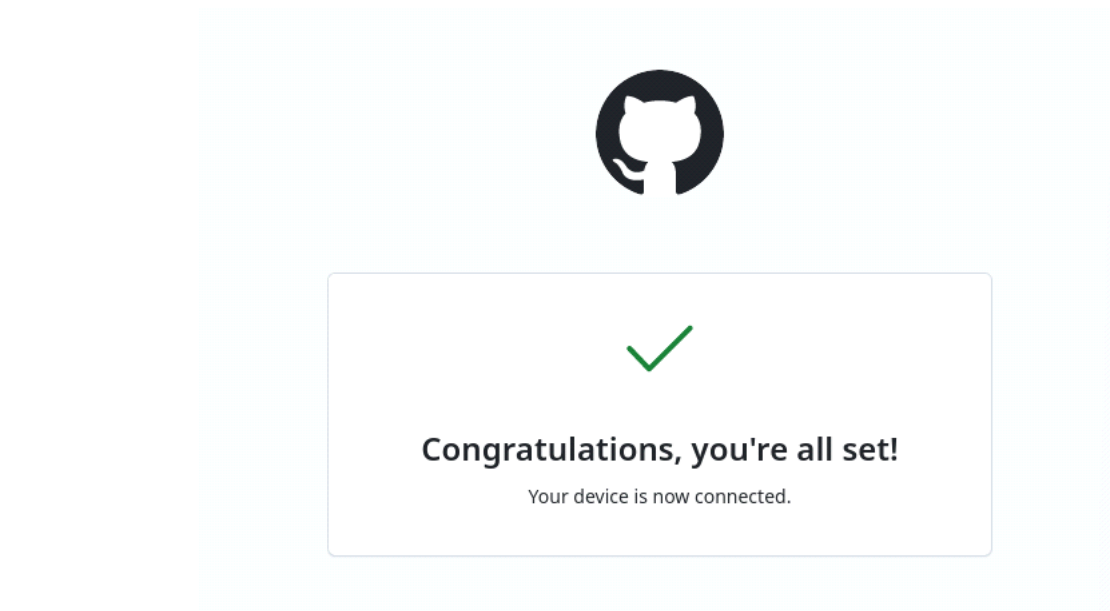


Рис. 3.18: Завершение авторизации через браузер

Вижу сообщение о завершении авторизации под именем evdvorkina (рис. 3.19).

```
! First copy your one-time code: EF62-D4A7
Press Enter to open github.com 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 mtuvshinbayar
```

Рис. 3.19: Завершение авторизации

- **Создание репозитория курса на основе шаблона**

Сначала создаю директорию с помощью утилиты `mkdir` и флага `-p`, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты `cd` перехожу в только что созданную директорию “Операционные системы”. Далее в терминале ввожу команду `gh repo create study_2022-2023_os-intro --template yamadharma/course-directory-student-`

template --public, чтобы создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе в директорию, я указываю ссылку с протоколом https, а не ssh, потому что при авторизации в gh выбрала протокол https (рис. 3.20).

```
liveuser@tmunkhjargal:~/work/study/2023-2024/Operating systems$ git clone --recursive https://github.com/mtuvshinbayar/study_2023-2024_oc-intro.git os-intro
Cloning into 'os-intro'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (33/33), 18.82 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharm/academic-presentation-markdown-template.git) registered for path 'template/presentation'
```

Рис. 3.20: Создание репозитория

Перехожу в каталог курса с помощью утилиты cd, проверяю содержание каталога с помощью утилиты ls (рис. 3.21).

```
liveuser@tmunkhjargal:~/work/study/2023-2024/Operating systems/os-intro$ ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE package.json README.git-flow.md  template
```

Рис. 3.21: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm, далее создаю необходимые каталоги используя makefile (рис. 3.22).

```
liveuser@tmunkhjargal:~/work/study/2023-2024/Operating systems/os-intro$ rm package.json
liveuser@tmunkhjargal:~/work/study/2023-2024/Operating systems/os-intro$ echo os-intro
> COURSE
liveuser@tmunkhjargal:~/work/study/2023-2024/Operating systems/os-intro$ make
```

Рис. 3.22: Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер (сохраняю добавленные изменения) с помощью команды git add и комментирую их с помощью git commit (рис. 3.23).

```
liveuser@tmunkhjargal:~/work/study/2023-2024/Operating systems/os-intro$ git add .
liveuser@tmunkhjargal:~/work/study/2023-2024/Operating systems/os-intro$ git commit -am 'feat(main): make course structure'
[master a04caf7] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
```

Рис. 3.23: Отправка файлов на сервер

Отправляю файлы на сервер с помощью git push (рис. 3.24).

```
liveuser@munkhjargal:~/work/study/2023-2024/Operating_systems/os-intro$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 950 bytes | 950.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mtuvshinbayar/study_2023-2024_oc-intro.git
  3b1aa98..a04caf7  master -> master
```

Рис. 3.24: Отправка файлов на сервер

• Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение по работе с git.

• Ответы на контрольные вопросы.

- Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо

изменение и т. д. VCS применяются для: Хранения полной истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.

- Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
- Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого

репозитория. В отличие от классических, в распределенных (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.

- Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
- Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в

хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.

- Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
- Создание основного дерева репозитория: `git init`

Получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull`

Отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push`

Просмотр списка изменённых файлов в текущей директории: `git status`

Просмотр текущих изменений: `git diff`

Сохранение текущих изменений: добавить все изменённые и/или созданные файлы и/или каталоги: `git add .`

добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов`

удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов`

Сохранение добавленных изменений:

сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'`

сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit`

создание новой ветки, базирующейся на текущей: `git checkout -b`

имя_ветки переключение на некоторую ветку: `git checkout имя_ветки`
(при переключении

на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)

отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки`

слияние ветки с текущим деревом: `git merge --no-ff имя_ветки`

Удаление ветки:

удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки`

принудительное удаление локальной ветки: `git branch -D имя_ветки`

удаление ветки с центрального репозитория: `git push origin :имя_ветки`

- `git push -all` отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
- Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
- Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

Список литературы

1. Лабораторная работа № 2 [Электронный ресурс] URL: <https://esystem.rudn.ru/mod/page/view>