

Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Зеленко Ирина Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Контрольные вопросы	16
5	Выводы	19

Список иллюстраций

3.1	Базовая настройка git	9
3.2	Создание ключа SSH	9
3.3	Создание ключа GPG	9
3.4	Ключ SSH создан	10
3.5	Ключ GPG создан	10
3.6	Отпечаток приватного ключа	11
3.7	Настройка подписей	11
3.8	Настройка gh	12
3.9	Создание репозитория	13
3.10	Настраиваем каталог курса	14
3.11	Отправляем наши файлы на сервер	15

Список таблиц

1 Цель работы

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

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

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.

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. # Выполнение лабораторной работы

Базовая настройка git:

1. Задаём имя и email владельца репозитория (1 и 2 строка на рисунке)
2. Настраиваем utf-8 в выводе сообщений git (3 строка на рисунке)
3. Настраиваем верификацию и подписание коммитов git. Зададим имя начальной ветки (будем называть её master) (4 строка на рисунке)
4. Параметр autocrlf (5 строка на рисунке)
5. Параметр safecrlf (6 строка на рисунке)


```

iyzelenko@dk2n22 ~ $ git config --global user.name "iyzelenko"
iyzelenko@dk2n22 ~ $ git config --global user.email "1132230297@pfur.ru"
iyzelenko@dk2n22 ~ $ git config --global core.quotepath false
iyzelenko@dk2n22 ~ $ git config --global init.defaultBranch master
iyzelenko@dk2n22 ~ $ git config --global core.autocrlf input
iyzelenko@dk2n22 ~ $ git config --global core.safecrlf warn

```

Рис. 3.1: Базовая настройка git

Создаём ключ SSH. В терминале вводим данную команду:

`ssh-keygen -t rsa -b 4096`

Далее во всех пунктах пользуемся клавишей Enter и получаем наш ключ.

```

iyzelenko@dk2n22 ~ $ 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/i/y/iyzelenko/.ssh/id_rsa):
/afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Qvk4S60A+z4QM1U6t6OD923aVWMViq9FD3AaBf4zYU0 iyzelenko@dk2n22
The key's randomart image is:
+---[RSA 4096]-----+
|  . . . . o.o E |
|  . . . . + o + |
|  .o + . . = = |
| +o + = . O . |
| .+ B S . = * |
| .o + * + o = |
| ..= o . . . |
|  o.o o.. |
|  .ooo |
+---[SHA256]-----+

```

Рис. 3.2: Создание ключа SSH

```

iyzelenko@dk2n22 ~ $ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:avCuJ/Cruk5ypQz23+h7s5o3OoPQEwy5/x94dWXBmM iyzelenko@dk2n22
The key's randomart image is:
+---[ED25519 256]---+
|  . |
|  .o |
|  .o . . |
|  .o o o . E |
| o+ . + S o o |
| +B+o = . . + |
| o.ooo... o o |
|  . o+==o |
|  .ooo00++ |
+---[SHA256]-----+

```

Рис. 3.3: Создание ключа GPG

Ключ нужно добавить на github. Для этого переходим на сайте в раздел


“Settings” и выбираем “SSH and GPG keys”.

SSH keys

[New SSH key](#)

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

Authentication keys

**irina**
SHA256: vFfhbU3y8w5oZsdI8phh230cDnJ4CCurXNWV6u0uucI
Added on Nov 6, 2023
Last used within the last 4 months — Read/write

Delete


Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

Рис. 3.4: Ключ SSH создан

GPG keys

[New GPG key](#)

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

**title**
Email address: zelenko.iren@yandex.ru
Key ID: FD015156D83303EC
Subkeys: 59EFB88DA01C01CF
Added on Feb 27, 2024

Delete

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

Рис. 3.5: Ключ GPG создан

Выводим список ключей и копируем отпечаток приватного ключа

```

iyzelenko@dk2n22 ~ $ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 2 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 2u
/afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.gnupg/pubring.kbx
-----
sec   rsa4096/FD015156D83303EC 2024-02-26 [SC]
      D882C4DFB4CA3023A2D969B0FD015156D83303EC
uid           [ абсолютно ] iyzelenko <zelenko.iren@yandex.ru>
ssb   rsa4096/59EFB88DA01C01CF 2024-02-26 [E]

sec   rsa4096/EB8117FB27FC7E7B 2024-02-27 [SC]
      0512A73AE98E5FB8A31645B9EB8117FB27FC7E7B
uid           [ абсолютно ] iyzelenko@dk2n22 <1132230297@pfur.ru>
ssb   rsa4096/54758ACDD5A738D3 2024-02-27 [E]

```

Рис. 3.6: Отпечаток приватного ключа

Настройка автоматических подписей коммитов git

```

-----BEGIN PGP PUBLIC KEY BLOCK-----
mQINPwoaQs0ITX18jxd39g1J/Jk/kRsaI0y18RLq7oav10EsID+p4FUYzppTmYNJg3
q00BLkT/XhvqTDb0rNKp4Ttsfsz12WiVyzneBx99qBXVEV5D3KVh2MbTNZvF4bv
I6wR0b3QZ8G6Y56s68cwidslAAoqH9/g1ZXcm4p34AB+Gdj3WQw21HfBz08mcA
1lxUE1LfFxv85Z6KLPrT/2ffj2TZhSgmZ8RNrbIDSoHB1Hd3VjS7GE+F866ees+w
kmOnNrffuke410tDGmIQ+591dosFtwGnUeyyH/UI76/1ECS4r6Y+iTbyTz5ZsA
JoBHOEEAyFBnrVv54kzcenSed4KtgAujgmh/NxdfYGY+c0SJfzdkhIrtlysNWUk9
IH7kPk1kZNSpckw4a7X1kqpn57gyBT/T83CmwSNY21vbYVSwLeRZCJHJE5wr8ASC
IQYOAvIz1Cy2v1eeI1834wzHJumCRp2JC0y28BrFb21pW9vsQJGkzLH5Qh+kLLK1
MxqfIH9KdDBhgHYMqmzSk97eIMBcgig0U+EW3dSLKg2ewXaBBNnPDDeg+c6Q==
=vvFLb
-----END PGP PUBLIC KEY BLOCK-----
iyzelenko@dk2n22 ~ $ git config --global user.signingkey FD015156D83303EC
iyzelenko@dk2n22 ~ $ git config --global commit.gpgsign true
iyzelenko@dk2n22 ~ $ git config --global gpg.program $(which gpg2)

```

Рис. 3.7: Настройка подписей

Возвращаемся в наш терминал и настраиваем gh командой:

gh auth login.

Во всех пунктах выбираем у(yes).

По полученной ссылке переходим в браузер на виртуальной машине и вводим код из терминала (находится перед ссылкой).

```
iyzelenko@dk2n22 ~ $ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_ed25519.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 245B-4CD8
Press Enter to open github.com in your browser...
[8157:8157:8227/125716.577596:ERROR:object_proxy.cc(576)] Failed to call method: org.freedesktop.ScreenSaver.GetActive: object_path= /org/freedesktop/ScreenSaver: org.freedesktop.DBus.Error.NotSupported: This method is not part of the idle inhibition specification: https://specifications.freedesktop.org/idle-inhibit-spec/latest/
[8157:8157:8227/125716.877498:ERROR:object_proxy.cc(576)] Failed to call method: org.gnome.ScreenSaver.GetActive: object_path= /org/gnome/ScreenSaver: org.freedesktop.DBus.Error.ServiceUnknown: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.Shell.ScreenShield was not provided by any .service files
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
✓ Authentication complete.
gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_ed25519.pub
✓ Logged in as iyzelenko
iyzelenko@dk2n22 ~ $
```

Рис. 3.8: Настройка gh

Создаём репозиторий курса на основе шаблона. Все нужные команды для создания были в указаниях к лабораторной работе. В 4 команде, вместо , указываем своё имя профиля на github.

1. `mkdir -p ~/work/study/2021-2022/“Операционные системы”`
2. `cd ~/work/study/2021-2022/“Операционные системы”`
3. `gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public`
4. `git clone --recursive git@github.com:/study_2021-2022_os-intro.git os-intro`

```

- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ SSH key already existed on your GitHub account: /afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/.ssh/id_ed
25519.pub
✓ Logged in as iyzelenko
! You were already logged in to this account
iyzelenko@dk2n22 ~ $ mkdir -p ~/work/study/2023-2024/"Операционные системы"/os-intro
iyzelenko@dk2n22 ~ $ cd ~/work/study/2023-2024/"Операционные системы"/os-intro
bash: d: команда не найдена
iyzelenko@dk2n22 ~ $ cd ~/work/study/2023-2024/"Операционные системы"/os-intro
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ gh repo create study_2022-2023_
os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ git clone --recursive git@github
:com:iyzelenko/study_2023-2024_arch--pc.git
fatal: целевой репозиторий 'study_2023-2024_arch--pc' уже существует и не является пустым каталогом.
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ cd ~/work/study/2023-2024/"Опер
ационные системы"/os-intro
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ echo os-intro > COURSE
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ ls
COURSE Makefile study_2023-2024_arch--pc
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make prepare
make: ./config/script/prepare: Нет такого файла или каталога
make: *** [Makefile:27: prepare] Ошибка 127
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make prepare
make: ./config/script/prepare: Нет такого файла или каталога
make: *** [Makefile:27: prepare] Ошибка 127
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make
make: ./config/script/prepare: Нет такого файла или каталога
make: *** [Makefile:27: prepare] Ошибка 127
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make
make: ./config/script/prepare: Нет такого файла или каталога
make: *** [Makefile:27: prepare] Ошибка 127
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $

```

Рис. 3.9: Создание репозитория

Настраиваем каталог курса. Для этого переходим в него командой:

```
cd ~/work/study/2021-2022/"Операционные системы"/os-intro
```

Далее командой `ls` проверяем, что мы в него перешли. В каталоге “os-intro” нам потребуется удалить файл “package.json”. Выполняем данную задачу командой:

```
rm package.json
```

Снова командой `ls` проверяем успешное выполнение удаления файла.

```

remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 КиБ | 1013.00 КиБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/afs/dk.sci.pfu.edu.ru/home/i/y/iyzelenko/work/study/2023-2024/Операционные системы/os-intro/study_2023-2024_arch--pc/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 КиБ | 2.17 МБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ffica72c60a304f24c'
Submodule path 'template/report': checked out '25e169d367953f60c76c251db299ed52852b401f'
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ cd ~/work/study/2023-2024/Операционные системы/os-intro
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ echo os-intro > COURSE
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make
make: ./config/script/prepare: Нет такого файла или каталога
make: *** [Makefile:27: prepare] Ошибка 127
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make repare
make: *** Нет правила для сборки цели «repare». Останов.
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make prepare
make: ./config/script/prepare: Нет такого файла или каталога
make: *** [Makefile:27: prepare] Ошибка 127
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ ^C
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ echo os-intro > COURSE
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $ make
make: ./config/script/prepare: Нет такого файла или каталога
make: *** [Makefile:27: prepare] Ошибка 127
iyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/os-intro $

```

Рис. 3.10: Настраиваем каталог курса

Создаём необходимые каталоги и отправляем наши файлы на сервер
make COURSE=os-intro

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

```

lyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/study_2023-2024_os-
intro $ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submodules

lyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/study_2023-2024_os-
intro $ git add .
lyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/study_2023-2024_os-
intro $ git commit -am 'feat(main): make course structure'
[master 944209e] feat(main): make course structure
1 file changed, 26 insertions(+)
 create mode 100644 config/script/Makefile
lyzelenko@dk2n22 ~/work/study/2023-2024/Операционные системы/study_2023-2024_os-
intro $ git push
Перечисление объектов: 7, готово.
Подсчет объектов: 100% (7/7), готово.
При скачке изменений используется до 6 потоков.
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 1.13 Киб | 1.13 Миб/с, готово.
Итого 4 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:lyzelenko/study_2023-2024_os-intro.git
   0b71007..944209e  master -> master

```

Рис. 3.11: Отправляем наши файлы на сервер

4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. История — место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.

6. Каковы основные задачи, решаемые инструментальным средством git?
Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
7. Назовите и дайте краткую характеристику командам git. git –version (Проверка версии Git) git init (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) git clone <https://www.github.com/username/repo-name> (Скопировать существующий удаленный Git-репозиторий) git remote (Просмотреть список текущих удалённых репозиториях Git) git remote -v (Для более подробного вывода) git add my_script.py (Можете указать в команде конкретный файл). git add . (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) git commit -am “Commit message” (Вы можете сжать все индексированные файлы и отправить коммит). git branch (Просмотреть список текущих веток можно с помощью команды branch) git –help (Чтобы узнать больше обо всех доступных параметрах и командах) git push origin master (Передать локальные коммиты в ветку удаленного репозитория).
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветки (branches)? Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы,

которые по какой-либо иной причине не должны попадать в коммиты.

5 Выводы

В ходе выполнения лабораторной работы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.