Титульный лист

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Лабораторная работа 2

По дисциплине "Операционные системы"

Выполнил:

Студент группы НПВбм-01-19

Студенческий билет №: 1032193844

Саидов Ахият Магомадович

Руководитель: Валиева Татьяна Рефатовна

Москва 2023

Цель работы

Мы изучим идеологию и применение средств контроля версий, также мы освоим умения по работе с git.

Начало работы

Настроим github.

Для этого предварительно создадим учетную страницу на сайте github.com.

Мы создали УС: https://github.com/Akhiyat; и заполнили необходимые данные.
Установим программное обеспечение.

Чтобы установить git из стандартного репозитория CentOS, используем менеджер пакетов yum.

```
[root@amsaidov ~]# sudo yum install git
Загружены модули: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.axelname.ru
 * extras: mirror.axelname.ru
 * updates: mirror.axelname.ru
Пакет git-1.8.3.1-24.el7_9.x86_64 уже установлен, и это последняя версия.
Выполнять нечего
```

Рисунок 1

Примечание: ранее уже был установлен git, исходя из этого, нам указывается, что устанавливать нечего.

Базовые настройки git.

Зададим имя и email владельца репозитория

```
[root@amsaidov ~]# git config --global user.name "AMSaidov"
[root@amsaidov ~]# git config --global user.email "akhiayts@gmail.com"
```

Рисунок 2

Настроим utf-8 в выводе сообщения git

```
[root@amsaidov ~]# git config --global core.quotepath false
```

Рисунок 3

Настроим верификацию и подписание коммитов git.

Зададим имя начальной ветки (будем называть ее master).

Параметр autocrlf.

Параметр safecrlf.

```
[root@amsaidov ~]# git config --global init.defaultBranch master
[root@amsaidov ~]# git config --global core.autocrlf input
[root@amsaidov ~]# git config --global core.safecrlf warn
```

Создадим ключи SSH.

По алгоритму rsa с ключем размеров 4096 бит.

По алгоритму ed25519.

```
[root@amsaidov ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+wMBSXQZEgKb4vUjJod5l7h05BIdwBU8epcil+uMaUQ root@amsaidov
The key's randomart image is:
+---[RSA 4096]----+
|  oo=**ooo       |
|   +.++o.        |
|. oo.oo..        |
|..+E*=.o.        |
| =.O+B+ S.       |
|  *.*.. ..       |
|  ..=   ..       |
|   + o   ..      |
|  .       ..     |
+----[SHA256]-----+
```

```
[root@amsaidov ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519.
Your public key has been saved in /root/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:wTOBUGhBK1h97YSUW54TugPU+qAV+e0o0vslKW2a9d0 root@amsaidov
The key's randomart image is:
+--[ED25519 256]--+
|  .o+Oo=.        |
| o  B.*.=.       |
|. .o.= O=o       |
|   .= + *+       |
|   + + +S.       |
|  o o.=..        |
|   ..o*..        |
|    .* + . .     |
|    o.. . . E    |
+----[SHA256]-----+
```

Рисунок 5

Создадим ключи GPG.

Генерируем ключ.

```
[root@amsaidov ~]# gpg --gen-key
gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог `/root/.gnupg'
gpg: создан новый файл настроек `/root/.gnupg/gpg.conf'
gpg: ВНИМАНИЕ: параметры в `/root/.gnupg/gpg.conf' еще не активны при этом запуске
gpg: создана таблица ключей `/root/.gnupg/secring.gpg'
gpg: создана таблица ключей `/root/.gnupg/pubring.gpg'
Выберите требуемый тип ключа:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (только для подписи)
   (4) RSA (только для подписи)
Ваш выбор (?-подробнее)? 1
ключи RSA могут иметь длину от 1024 до 4096 бит.
Какой размер ключа необходим? (2048) 4096
Запрашиваемый размер ключа 4096 бит
Выберите срок действия ключа.
         0 = без ограничения срока действительности
      <n>  = срок действительности n дней
      <n>w = срок действительности n недель
      <n>m = срок действительности n месяцев
      <n>y = срок действительности n лет
Ключ действителен до? (0) 0
Ключ не имеет ограничения срока действительности
Все верно? (y/N) y

GnuPG необходимо составить UserID в качестве идентификатора ключа.

Ваше настоящее имя: AMSaidov
Email-адрес: akhiayts@gmail.com
Комментарий:
Вы выбрали следующий User ID:
    "AMSaidov <akhiayts@gmail.com>"

Сменить (N)Имя, (C)Комментарий, (E)email-адрес или (O)Принять/(Q)Выход? O
Для защиты секретного ключа необходима фраза-пароль.
```

– 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. – Комментарий.

Рисунок 7

Добавление GPG ключа в GitHub.

Скопируем наш сгенерированный PGP ключ в буфер обмена.

```
[root@amsaidov ~]# gpg --armor --export 0C41681B013B92AF9AE2D307AAEB10C3909663DB
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)

mQINBGRzxmUBEADUVjauNjTmKzVgfIfKzvFQbCD1+QAmIoTUZfqq3mfcDE755dWd
+NcHpaTA8XQVJ8RerD+m0trXstkFwMMYUKpQ4OGLOaRygU/JtIzdHIDIg6M3Oq19
0ZrySp7WosMDMJqak0KJmOvN6HuGlzrluChq8FzEKry72ZCVxcphvn31/bghdYmw
VX1yZjDs3RlVdNkUVoPivUczdsKK/M03bzGbaQQBXDh/c4p7u67LuUKMIk/97efp
oHZKGSr4WlknMXbKbWf1dHi8Pgg/VsMryKa8WHgeo+Ipbd+RMFsyE0U1QXcO9bw6
fBcc6ugowKXulffgQoqPiJQWRSu+OQOHtcqh5JdYGhcDfYt9QmlZrP/BIFxnpzYp
/UKU4ZDp0+ez4077RaOLPF66Z8xg1rtIAbfimWrlA8Jqovcr+W4VOtmE/RQHyp6K
FyHcfAM3KR0kKan/Q8XBtxpIIN55ZxMpHem5mhoJs16OL3dsZgPkoiOtRaWC4Ih5
A4M2+8anmFzs9A3iD1d7Q1+yILIQkWfxwjmW7dStYjNipYXX13+d6xJ10ZsD6jTA
cbaZAi13S7BU7HXKf3M0SzaDBPas6aJeX67hqXpLhtZdN0Cg3LFS/9vQtycjeBUu
IRXdiY0FlpMAOOISnUPEF9DdF4f8PADWFnumulZuXQuKK8Q9PI1M7pGqIwARAQAB
tB1BTVNhaWRvdiA8YWtoaWF5dHNAZ21haWwuY29tPokCOQQTAQIAIwUCZHPGZQIb
AwcLCQgHAwIBBhUIAgkKCwQWAgMBAh4BAheAAAoJEKrrEMOQlmPbfR0P/jDDgY05
oMZCL9d8rOyqfJsL7TvrOiooAuFFBujw4XyBIOMFsYw8NI/i2j3FDaW5NLu8F+jj
SHwJnIU4LKa1bjcnfbjEqhpWCje+jda1wl5MgmM5z6Tq2499LcrgR5s2xXU0tHeY
OhcclqRnlKxZbx7fwuUhYrtSb+TPnjJ1NRn9TXQSaqm37w2vjV7V0YRsnuJl3dXK
zUUBIEG4dRk3FqxKGdZ4HaZegbwRGRW4heHqSwIsUxPoh9lzVpA8noas/gmlPnL4
bffK7b2ZjUScSEXiJYwWW3lB/RkkF0hj4XTDsisQngQE7tYaWuRsJJ7I+u6ZwzZ/
GD6/Cl+H2D8Ib6EJSWjMUy+EQHUW9LY/BxRaplciKZ/gS++lqcRvduRnlrffmhod
```

Рисунок 8

Вставим ключ в GitHub.

Рисунок 9

Используем введенный email, укажем Git применять его при подписи коммитов.

```
[AMSaidov@amsaidov ~]$ git config --global user.signingkey 0C41681B024B92AF9AE2D
307AAEB10C3909663DB
[AMSaidov@amsaidov ~]$ git config --global commit.gpgsign true
[AMSaidov@amsaidov ~]$ git config --global gpg.program $(which gpg2)
```

Рисунок 10

Создадим репозиторий курса.

```
[AMSaidov@amsaidov ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[AMSaidov@amsaidov ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[AMSaidov@amsaidov Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/
course-directory-student-template --public
✓ Created repository Akhiyat/study_2022-2023_os-intro on GitHub
[AMSaidov@amsaidov Операционные системы]$ git clone --recursive git@github.com:Akhiyat/study_2022-2023_o
```

Рисунок 11

Перейдем в каталог курса.

```
[AMSaidov@amsaidov Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
```

Рисунок 12

Удалим лишние файлы.

```
[AMSaidov@amsaidov os-intro]$ rm package.json
```

Рисунок 13

Создадим необходимые каталоги.

```
[AMSaidov@amsaidov os-intro]$ echo os-intro > COURSE
[AMSaidov@amsaidov os-intro]$ make
```

Рисунок 14

Отправим файлы на сервер.

```
[AMSaidov@amsaidov os-intro]$ git add .
[AMSaidov@amsaidov os-intro]$ git commit -am 'feat(main): make course structure'
[AMSaidov@amsaidov os-intro]$ git push
```

Рисунок 15

Вывод

Мы изучили идеологию и применение средств контроля версий. Мы освоили умения по работе с git.

Контрольные вопросы

Контрольные вопросы.

1. 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
2. - Хранилище (репозиторий) – это система, которая обеспечивает хранение всех существовавших версий файлов.
   - Commit - запись изменений.
   - История - список предыдущих изменений.
   - Рабочая копия – копия файла, с которой непосредственно ведётся работа (находится вне репозитория) С помощью коммитов изменения, внесённые в рабочую копию, заносятся в хранилище. Благодаря истории можно отследить изменения, вносимые в репозиторий. Перед началом работы рабочую копию можно получить из одной из версий, хранящихся в репозитории.
3. В централизованных СКВ все файлы хранятся в одном репозитории, и каждый пользователь может вносить изменения. В децентрализованных их несколько, и они могут обмениваться изменениями между собой, а центрального репозитория может не существовать вообще. Среди классических (т.е. централизованных) VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial.
4. Получить нужную версию проекта (рабочую копию), внести в неё необходимые изменения, сделать нужный коммит, создав при этом новую версию проекта (старые не удаляются).
5. Аналогично единоличной работе, но также можно объединить внесённые разными пользователями изменения, отменить изменения или заблокировать некоторые файлы для изменения, обеспечив привилегированный доступ конкретному разработчику.

6. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Git позволяет создавать локальные репозитории и вносить в них изменения, а также работать с удалёнными репозиториями.
7. - 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 5)просмотр текущих изменения: git diff
   - сохранение текущих изменений: а)добавить все изменённые и/или созданные файлы и/или каталоги: git add . б)добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена*файлов в)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: а)сохранить все добавленные изменения и все изменённые файлы: git commit -am 'Описание коммита' б)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
8. Допустим, нужно добавить в проект новый файл file.txt Загрузим нужную версию из удалённого репозитория: git checkout last (last – имя нужной нам ветки) Добавим файл в локальный репозиторий: git add file.txt (файл лежит в том же каталоге, что и репозиторий) Сохраним изменения: git commit –am “file.txt was added” Отправим изменения в удалённый репозиторий: git push
9. СКВ могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Это удобно при работе над одним проектом нескольких человек, или если вносимые на каждой из ветвей изменения будут разительно отличаться (например, создание программ с разным функционалом на базе одного интерфейса).
10. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять впоследствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы