

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**  
**Факультет физико-математических и естественных наук**  
**Кафедра прикладной информатики и теории вероятностей**

Отчет  
По лабораторной работе №2  
Дисциплина: операционные системы

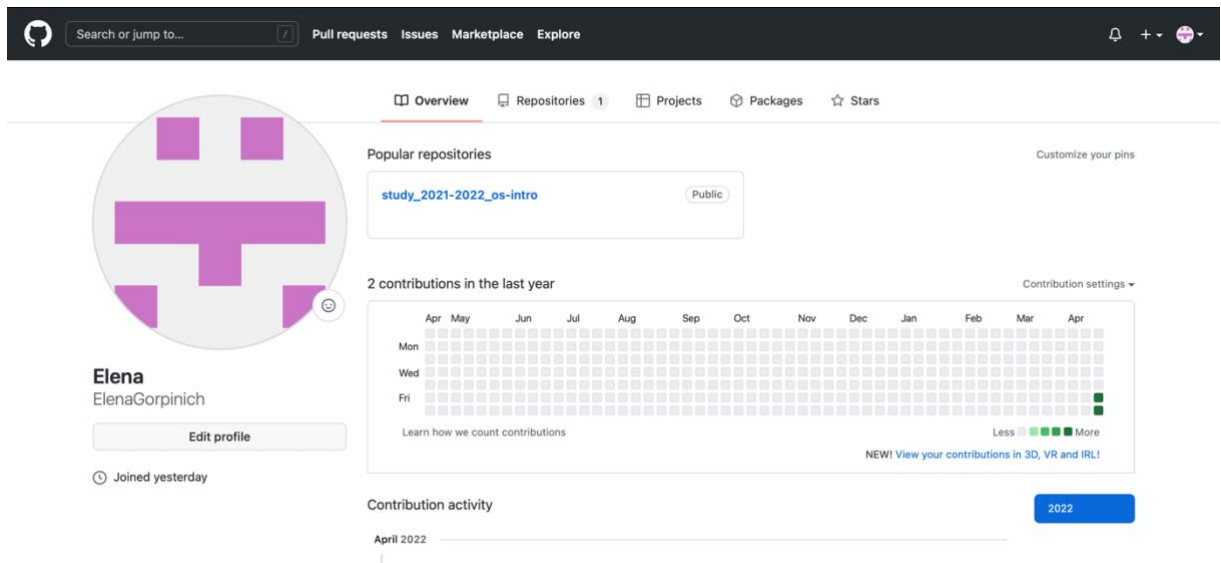
Студентка: Горпинич Елена Михайловна  
Группа: НПМбд-01–21

**Москва**  
**2022**

Цель работы: изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

Ход работы:

- 1) Настройка GitHub. Изначально создаём учетную запись и заполняем основные данные на официальном сайте. <https://github.com>.



- 2) Затем синхронизируем учетную запись с компьютером с помощью терминала.

```
emgorpinich@emgorpinich:~$ git config --global user.name "Elena Gorpinich"
emgorpinich@emgorpinich:~$ git config --global user.email "gorpinich-lena@mail.ru"
```

- 3) Начинаем базовую настройку git: настроим utf-8 в выводе сообщений git, верификацию и подписание коммитов git, зададим имя начальной ветки (будем называть её master), параметр autocrlf, параметр safecrlf

```
emgorpinich@emgorpinich:~$ git config --global core.quotepath false
emgorpinich@emgorpinich:~$ git config --global init.defaultBranch master
emgorpinich@emgorpinich:~$ git config --global core.autocrlf input
emgorpinich@emgorpinich:~$ git config --global core.safecrlf warn
```

- 4) Создадим ключ ssh по алгоритму rsa с ключевым размером 4096 бит

```
emgorpinich@emgorpinich:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/emgorpinich/.ssh/id_rsa): lab2_1
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in lab2_1
Your public key has been saved in lab2_1.pub
The key fingerprint is:
SHA256:C4qjXaz9nBcwua33lXF1skiX1LSyB/BFk8bCHBKl1RY emgorpinich@emgorpinich
The key's randomart image is:
+---[RSA 4096]-----+
|           o+=+E=|
|           +=00+|
|          .  ..+B.+|
|         +   . o++o|
|        .=S   o o+ |
|       o ...o.  +.  |
|      o +   ... o   |
|     o =   ...o .   |
|    . o ..+o ..    |
+---[SHA256]-----+
```

- 5) Создадим ключ pgr. Генерируем ключ и из предложенных опций выбираем опции описанные в условии лабораторной работы.

```
emgorpinich@emgorpinich:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 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.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (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
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: ElenaGorpinich
Email address: gorpinich-lena@mail.ru
Comment:
You selected this USER-ID:
    "ElenaGorpinich <gorpinich-lena@mail.ru>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 46A3B8E79662EF24 marked as ultimately trusted
gpg: revocation certificate stored as '/home/emgorpinich/.gnupg/openpgp-revocs.d/75F4D4D859A5758BADDECA5E46A3B8E79662EF24.rev'
public and secret key created and signed.

pub   rsa4096 2022-04-23 [SC]
      75F4D4D859A5758BADDECA5E46A3B8E79662EF24
uid           ElenaGorpinich <gorpinich-lena@mail.ru>
sub   rsa4096 2022-04-23 [E]
```

- 6) Добавим PGP ключ в GitHub. Для этого выводим список ключей и скопируем отпечаток приватного ключа. Скопируем наш сгенерированный PGP ключ в буфер обмена. С помощью настроек GitHub (<https://github.com/settings/keys>), добавим наш скопированный ключ, вставив его в поле ввода, которое появилось после нажатия кнопки New GPG key

```
emgorpinich@emgorpinich:~$ gpg --list-secret-keys --keyid-format LONG
/home/emgorpinich/.gnupg/pubring.kbx
-----
sec   rsa4096/46A3B8E79662EF24 2022-04-23 [SC]
      75F4D4D859A5758BADDECA5E46A3B8E79662EF24
uid   [ultimate] ElenaGorpinich <gorpinich-lena@mail.ru>
ssb   rsa4096/A91F019269262B4D 2022-04-23 [E]

emgorpinich@emgorpinich:~$ gpg --armor --export 46A3B8E79662EF24 | xclip -sel clip
```

## PGP keys / Add new

### Key


Begins with '-----BEGIN PGP PUBLIC KEY BLOCK-----'

Add GPG key

## PGP keys

New GPG key

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



**Email address:** gorpinich-lena@mail.ru  
**Key ID:** 46A3B8E79662EF24  
**Subkeys:** A91F019269262B4D  
 Added on 23 Apr 2022

PGP
Delete

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

### 7) Настроим автоматических подписей коммитов git

```
emgorpinich@emgorpinich:~$ git config --global user.signingkey 46A3B8E79662EF24
emgorpinich@emgorpinich:~$ git config --global commit.gpgsign true
emgorpinich@emgorpinich:~$ git config --global gpg.program $(which gpg2)
```

### 8) Создадим новый репозиторий курса на основе шаблона и далее настроим каталог курса, удалив ненужные файлы, создав необходимые каталоги и отправив файлы на сервер

# Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner \*

 ElenaGorpinich ▾

Repository name \*

study\_2021-2022\_os-intro ✓

Great repository names are short and lowercase. Your new repository will be created as `study_2021-2022_os-intro-arnacle?`

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

## Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more](#).

## Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: None ▾


## Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

License: None ▾

 You are creating a public repository in your personal account.

Create repository



**Elena**  
ElenaGorpinich  
[Edit profile](#)

[Overview](#) [Repositories 1](#) [Projects](#) [Packages](#) [Stars](#)

Type ▾ Language ▾ Sort ▾

[New](#)

[study\\_2021-2022\\_os-intro](#) Public

☆ Star ▾

Updated 1 hour ago



Вывод:

В данной лабораторной работе я научилась работать с Github (создавать и привязывать учетную запись к компьютеру). Разобрала основные команды git и рассмотрела, как их применять при работе с Github. Изучила идеологию и научилась применять средства контроля версий

Контрольные вопросы:

- VCS (Система контроля версий) — 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.
- Терминология VCS:
  - Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией.
  - Commit сохранение изменений в репозитории
  - Рабочая копия - копия проекта, связанная с репозиторием

- *Централизованные VCS*

Одно основное хранилище всего проекта

Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно (Subversion, CVS, TFS, VAULT, AccuRev)

*Децентрализованные VCS*

У каждого пользователя свой вариант (возможно, не один) репозитория  
Присутствует возможность добавлять и забирать изменения из любого репозитория. (Git, Mercurial, Bazaar)

- Поддержка автономной работы; локальные фиксации изменений могут быть отправлены позже  
Каждое рабочее дерево в git содержит хранилище с полной историей проекта  
Ни одно хранилище не является по своей природе более важным, чем любое другое  
Быстрое и легкое ветвление
- При выполнении команды git –help можно увидеть список команд git, используемых в различных ситуациях

These are common Git commands used in various situations:

start a working area (see also: `git help tutorial`)

<code>clone</code>	Clone a repository into a new directory
<code>init</code>	Create an empty Git repository or reinitialize an existing one

work on the current change (see also: `git help everyday`)

<code>add</code>	Add file contents to the index
<code>mv</code>	Move or rename a file, a directory, or a symlink
<code>restore</code>	Restore working tree files
<code>rm</code>	Remove files from the working tree and from the index
<code>sparse-checkout</code>	Initialize and modify the sparse-checkout

examine the history and state (see also: `git help revisions`)

<code>bisect</code>	Use binary search to find the commit that introduced a bug
<code>diff</code>	Show changes between commits, commit and working tree, etc
<code>grep</code>	Print lines matching a pattern
<code>log</code>	Show commit logs
<code>show</code>	Show various types of objects
<code>status</code>	Show the working tree status

grow, mark and tweak your common history

<code>branch</code>	List, create, or delete branches
<code>commit</code>	Record changes to the repository
<code>merge</code>	Join two or more development histories together
<code>rebase</code>	Reapply commits on top of another base tip
<code>reset</code>	Reset current HEAD to the specified state
<code>switch</code>	Switch branches
<code>tag</code>	Create, list, delete or verify a tag object signed with GPG

collaborate (see also: `git help workflows`)

<code>fetch</code>	Download objects and refs from another repository
<code>pull</code>	Fetch from and integrate with another repository or a local branch
<code>push</code>	Update remote refs along with associated objects

- git fetch

Команда `git fetch` связывается с удалённым репозиторием и забирает из него все изменения, которых у вас пока нет и сохраняет их локально.

Мы познакомились с ней в разделе Получение изменений из удалённого репозитория — Fetch и Pull главы 2 и продолжили знакомство в разделе Удалённые ветки главы 3.

Мы использовали эту команду в нескольких примерах из раздела Участие в проекте.

Мы использовали её для скачивания запросов на слияние (pull request) из других репозиториев в разделе Ссылки на запрос слияния главы 6, также мы рассмотрели использование `git fetch` для работы с упакованными репозиториями в разделе Создание пакетов главы 7.

Мы рассмотрели тонкую настройку `git fetch` в главе и Спецификации ссылок.

#### git pull

Команда `git pull` работает как комбинация команд `git fetch` и `git merge`, т. е. Git вначале забирает изменения из указанного удалённого репозитория, а затем пытается слить их с текущей веткой.

Мы познакомились с ней в разделе Получение изменений из удалённого репозитория — Fetch и Pull главы 2 и показали как узнать, какие изменения будут приняты в случае применения в разделе Просмотр удаленного репозитория главы 2.

Мы также увидели как она может оказаться полезной для разрешения сложностей при перемещении веток в разделе Меня базу, меняй основание главы 3.

Мы показали как можно использовать только URL удалённого репозитория без сохранения его в списке удалённых репозиториях в разделе Извлечение удалённых веток главы 5.

И наконец мы показали как проверять криптографические подписи полученных коммитов, используя опцию `--verify-signatures` в разделе Подпись коммитов главы 7.

`git push`

Команда `git push` используется для установления связи с удалённым репозиторием, вычисления локальных изменений отсутствующих в нём, и собственно их передачи в вышеупомянутый репозиторий. Этой команде нужно право на запись в репозиторий, поэтому она использует аутентификацию.

Мы познакомились с этой командой в разделе Отправка изменений в удалённый репозиторий (Push) главы 2. Там мы рассмотрели основы обновления веток в удалённом репозитории. В разделе Отправка изменений главы 3 мы подробнее познакомились с этой командой, а в разделе Отслеживание веток главы 3 мы узнали как настроить отслеживание веток для автоматической передачи на удалённый репозиторий. В разделе Удаление веток на удалённом сервере главы 3 мы использовали флаг `--delete` для удаления веток на сервере, используя `git push`.

На протяжении раздела Участие в проекте мы показали несколько примеров использования `git push` для совместной работы в нескольких удалённых репозиториях одновременно.

В разделе Публикация изменений подмодуля главы 7 мы использовали опцию `--recurse-submodules` чтобы удостовериться, что все подмодули будут опубликованы перед отправкой на сервера проекта, что может быть реально полезным при работе с репозиториями, содержащими подмодули.

В разделе Прочие хуки на стороне клиента главы 8 мы поговорили о триггере `pre-push`, который может быть выполнен перед отправкой данных, чтобы проверить возможность этой отправки.

Наконец, в разделе Спецификации ссылок для отправки данных на сервер главы 10 мы рассмотрели передачу данных с полным указанием передаваемых ссылок, вместо использования распространённых сокращений. Это может быть полезным если вы хотите очень точно указать, какими изменениями хотите поделиться.

`git remote`

Команда `git remote` служит для управления списком удалённых репозиториях. Она позволяет сохранять длинные URL репозиториях в виде понятных коротких строк, например «`origin`», так что вам не придётся забивать голову всякой ерундой и набирать её каждый раз для



связи с сервером. Вы можете использовать несколько удалённых репозиториях для работы и `git remote` поможет добавлять, изменять и удалять их.

Эта команда детально рассмотрена в разделе Работа с удалёнными репозиториями главы 2, включая вывод списка удалённых репозиториях, добавление новых, удаление или переименование существующих.

Она используется практически в каждой главе, но всегда в одном и том же виде: `git remote add <имя> <URL>`.

`git archive`

Команда `git archive` используется для упаковки в архив указанных коммитов или всего репозитория.

Мы использовали `git archive` для создания тарбола (`tar.gz` файла) всего проекта для передачи по сети в разделе Подготовка релиза главы 5.

`git submodule`

Команда `git submodule` используется для управления вложенными репозиториями. Например, это могут быть библиотеки или другие, используемые не только в этом проекте ресурсы. У

команды `submodule` есть несколько подкоманд — `add`, `update`, `sync` и др. — для управления такими репозиториями.

- Ветвь – независимое направление разработки. Изменения, вносимые в одну ветвь, не влияют на остальные ветви. Ветви используются, например, для внесения изменений, которые могут дестабилизировать код или для поддержки релизов. Изменения могут быть перенесены между ветвями посредством слияния.

- Вы можете заставить Git игнорировать определенные файлы и каталоги, то есть исключить их от отслеживания Git - путем создания одного или нескольких файлов `.gitignore` в вашем репозитории.

В проектах программного обеспечения `.gitignore` обычно содержит список файлов и / или каталогов, которые генерируются во время процесса сборки или во время выполнения. Записи в файле `.gitignore` могут включать имена или пути, указывающие на: временные ресурсы, например, кэши, файлы журналов, скомпилированный код и т. д.

файлы локальной конфигурации, которые не должны использоваться совместно с другими разработчиками

файлы, содержащие секретную информацию, такие как пароли входа, ключи и учетные данные

При создании в каталоге верхнего уровня правила будут применяться рекурсивно ко всем файлам и подкаталогам во всем репозитории. При создании в подкаталоге правила будут применяться к этому конкретному каталогу и его подкаталогам.

Когда файл или каталог игнорируются, это не будет: отслеживается Git

сообщается командами, такими как `git status` или `git diff` с такими командами, как `git add -A`

В необычном случае, когда вам нужно игнорировать отслеживаемые файлы, следует соблюдать особую осторожность