

# **Шаблон отчёта по лабораторной работе 2**

**Архитектура компьютера**

Волчкова Елизавета Дмитриевна

# Содержание

<b>1</b>	<b>Цель работы.</b>	<b>5</b>
<b>2</b>	<b>Задание для самостоятельной работы.</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение.</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы.</b>	<b>9</b>
<b>5</b>	<b>Вывод.</b>	<b>17</b>
<b>6</b>	<b>Список литературы.</b>	<b>18</b>

# Список иллюстраций

4.1	1lab03 . . . . .	9
4.2	2lab03 . . . . .	10
4.3	3lab03 . . . . .	10
4.4	4lab03 . . . . .	11
4.5	5lab03 . . . . .	11
4.6	6lab03 . . . . .	12
4.7	11lab03 . . . . .	14
4.8	12lab03 . . . . .	15
4.9	13lab03 . . . . .	15
4.10	14lab03 . . . . .	16

## **Список таблиц**

# **1 Цель работы.**

Необходимо понять и изучить идеологию и применение средств контроля версий, а также приобрести практические навыки по работе с системой git.

## **2 Задание для самостоятельной работы.**

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab02>report).
2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
3. Загрузите файлы на github

### 3 Теоретическое введение.

Системы контроля версий. Общие понятия. 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. Демидова А. В. 14 Архитектура ЭВМ 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 6 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



## 4 Выполнение лабораторной работы.

Я начинаю с проверки и получения изменений из центрального репозитория : `git checkout master` `git pull` `git checkout -b имя_ветки`. Затем вношу изменения в локальном дереве или ветке Далее я разместила их в центральном репозитории. Для этого я проверила, какие файлы изменились к текущему моменту: `git status` и удалила лишние файлы. Затем посмотрела текст изменений на предмет соответствия правилам ведения чистых коммитов: `git diff`

```
root@LAPTOP-KE7VHG2Q:~# git add /root/labs
root@LAPTOP-KE7VHG2Q:~# git checkout master
error: pathspec 'master' did not match any file(s) known to git
root@LAPTOP-KE7VHG2Q:~# git checkout -b labs
Switched to a new branch 'labs'
root@LAPTOP-KE7VHG2Q:~# git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=<remote>/<branch> labs

root@LAPTOP-KE7VHG2Q:~# git diff
root@LAPTOP-KE7VHG2Q:~#
```

Рис. 4.1: 1lab03

```

root@LAPTOP-KE7VHG2Q:~# git add labs
root@LAPTOP-KE7VHG2Q:~# git status
On branch Liza

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   labs/firsname.txt
    new file:   labs/id-group.txt
    new file:   labs/lastname.txt

Untracked files:

```

Рис. 4.2: 2lab03

Используя команды добавления и/или удаления с нужными опциями: `git add имена_файлов` `git rm имена_файлов` я пометила файлы, которые не должны попасть в комитет.

```

root@LAPTOP-KE7VHG2Q:~# git add /root/labs/
root@LAPTOP-KE7VHG2Q:~# git rm /root/labs/
fatal: not removing '/root/labs/' recursively without -r
root@LAPTOP-KE7VHG2Q:~# git rm -r /root/labs/
error: the following files have changes staged in the index:
    labs/firsname.txt
    labs/id-group.txt
    labs/lastname.txt

```

Рис. 4.3: 3lab03

Git add позволяет сохранить все файлы в каталоге . Затем сохранила изменения, поясняя, что было сделано: `git commit -am "Some commit message"` и отправила в центральный репозиторий: `git push origin имя_ветки` или `git push`.

```

root@LAPTOP-KE7VHG2Q:~# git commit -am "All right!"
[labs (root-commit) c939eaa] All right!
Committer: root <root@LAPTOP-KE7VHG2Q>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/firsnome.txt
create mode 100644 labs/id-group.txt
create mode 100644 labs/lastname.txt
root@LAPTOP-KE7VHG2Q:~#

```

Рис. 4.4: 4lab03

```

root@LAPTOP-KE7VHG2Q:~# git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

    git remote add <name> <url>

and then push using the remote name

    git push <name>

root@LAPTOP-KE7VHG2Q:~#

```

Рис. 4.5: 5lab03

Сначала я сделала предварительную конфигурацию git, затем открыла терминал и ввела следующие команды, указав имя и email владельца репозитория: `git config --global user.name ""` `git config --global user.email "work@mail"` Далее настроила utf-8 в выводе сообщений git: `git config --global core.quotePath false`. Потом задавала имя начальной ветки (master): `git config --global init.defaultBranch master`.

Параметр autocrlf: git config --global core.autocrlf input  
Параметр safecrlf: git config --global core.safecrlf warn

```
root@LAPTOP-KE7VHG2Q:~# git config --global user.name "Liza"
root@LAPTOP-KE7VHG2Q:~# git config --global user.email "liza.volchkova.s7@gmail.com"
root@LAPTOP-KE7VHG2Q:~# git config --global core.quotepath false
root@LAPTOP-KE7VHG2Q:~# git config --global init.defaultBranch maste
root@LAPTOP-KE7VHG2Q:~# git config --global core.autocrlf input
root@LAPTOP-KE7VHG2Q:~# git config --global core.safecrlf warn
root@LAPTOP-KE7VHG2Q:~#
```

Рис. 4.6: 6lab03

Для идентификации пользователя на сервере репозитория cutythbhjdfkf па-  
ру ключей (приватный и открытый): ssh-keygen -C "Liza2006- ux liza.volchkova.s7@gmail.com".

```
root@LAPTOP-KE7VHG2Q:~# ssh-keygen -C "Liza2006- ux liza.volchkova.s7@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): Liza2006
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in Liza2006
Your public key has been saved in Liza2006.pub
The key fingerprint is:
SHA256:rmEK6F7AMBVEXOR1AB71iu9njIrmKyyExI9LWh7A8w Liza2006- ux liza.volchkova.s7@gmail.com
The key's randomart image is:
+----[RSA 3072]-----+
|  o*o+o==          |
|  . . +...         |
| =o .. o .         |
| =E+ .o . .        |
| .+oo . S          |
| oo.o  o .         |
| ooo+. o o         |
| + =.oo+ .         |
| =*  ..o+o         |
+----[SHA256]-----+
root@LAPTOP-KE7VHG2Q:~# 123456789
```


Welcome to GitHub!

Let's begin the adventure

Enter your email\*

✓ liza.volchkova.s7@gmail.com

Create a password\*

→  

-----

Password is strong

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

Welcome to GitHub!

Let's begin the adventure

Enter your email\*

✓ liza.volchkova.s7@gmail.com

Create a password\*

✓ .....

Enter a username\*

→



## SSH keys

There are no SSH keys associated with your account.

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

## GPG keys

There are no GPG keys associated with your account.

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

## Vigilant mode

### ☐ Flag unsigned commits as unverified

This will include any commit attributed to your account but not signed with your GPG or S/MIME key. Note that this will include your existing unsigned commits.

[Learn about vigilant mode.](#)

Потом загрузила сгенерённый открытый ключ. Зашла на сайт <http://github.org/> под своей учётной записью и перешла в меню Setting. После этого выбрала в боковом меню SSH and GPG keys и нажала кнопку New SSH key. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставила ключ в появившееся на сайте поле и указвала для ключа имя (Title).

Открыла терминал и создала каталог для предмета «Архитектура компьютера»: `mkdir -p ~/work/study/2023-2024/«Архитектура компьютера»`.

```
root@LAPTOP-KE7VHG2Q:~# mkdir -p ~/work/study/2024-2025/"Архитектура компьютера"
root@LAPTOP-KE7VHG2Q:~# cd ~/work/study/2024-2025/"Архитектура компьютера"
```

Рис. 4.7: 11lab03

Репозиторий на основе шаблона создала через web-интерфейс github. Перейдя на страницу репозитория с шаблоном курса <https://github.com/yamadharma/course-directory-student-template>.

Далее выбрала Use this template, в открывшемся окне задала имя репозитория

(Repository name) study\_2023–2024\_arhpc и создала репозиторий (кнопка Create repository from template).

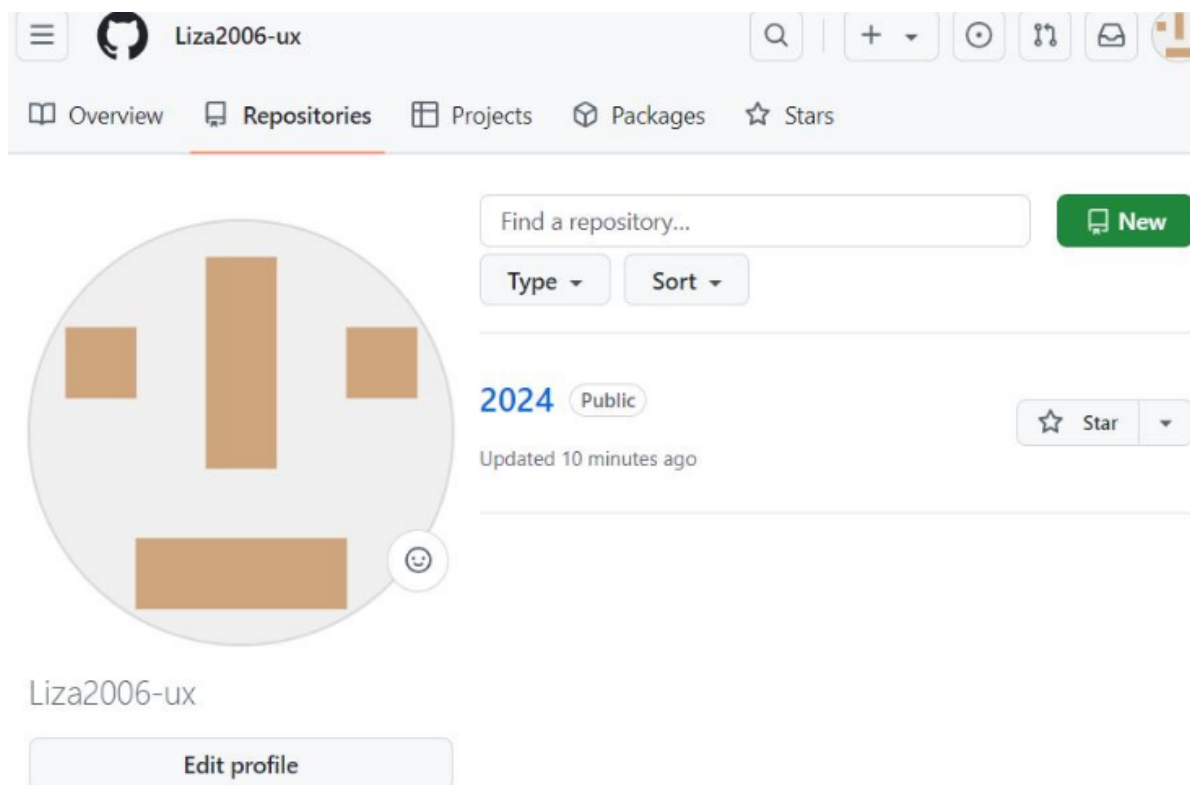


Рис. 4.8: 12lab03

Открыла терминал и перешла в каталог курса: `cd ~/work/study/2023–2024/“Архитектура компьютера”` клонировала созданный репозиторий: `git clone --recursive git@github.com:/study_2023–2024_arh-pc.git` ↪ arch-pc Ссылку для клонирования скопировала на странице созданного репозитория Code -> SSH:

```
root@LAPTOP-KE7VHG2Q:~/work/study/2024-2025/Архитектура компьютера# git clone --recursive git@github.com:Liza2006-ux/2024-2025.git arch-pc
fatal: bad boolean config value 'warn' for 'core.safecrlf'
root@LAPTOP-KE7VHG2Q:~/work/study/2024-2025/Архитектура компьютера# cd ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc
-bash: cd: /root/work/study/2024-2025/Архитектура компьютера/arch-pc: No such file or directory
root@LAPTOP-KE7VHG2Q:~/work/study/2024-2025/Архитектура компьютера# rm package.json
```

Рис. 4.9: 13lab03

Потом перешла в каталог курса: `cd ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc`, затем удалила лишние файлы: `rm package.json`

```
root@LAPTOP-KE7VHG2Q:~/work/study/2024-2025/Архитектура компьютера# git commit -am 'feat(main)  
make course structure'  
fatal: bad boolean config value 'warn' for 'core.safecrlf'
```

Рис. 4.10: 14lab03

Создала необходимые каталоги: `echo arch-pc > COURSE make`, отпраивла файлы на сервер: `git add . git commit -am 'feat(main): make course structure' git pus`, потом проверила правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.



## **5 Вывод.**

Целью работы являлось - изучение идеологии и применение средств контроля версий и приобрести практических навыков по работе с системой git. В процессе работы я использовала разные методы и техники. Так, я разобралась и изучила на практике необходимые команды.

## 6 Список литературы.

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784398337.
9. — ISBN 9781784398337.
10. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
11. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).

16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер,
17. — 874 с. — (Классика Computer Science). 16
18. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб.  
: Питер, 2015. — 1120 с. — (Классика Computer Science)