Лабораторная работа N3.

Язык разметки Markdown

Соловьев Богдан Михайлович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

2 Задание

1. Настройка GitHub
2. Базовая настройка Git
3. Создание SSH-ключа
4. Создание рабочего пространства и репозитория курса на основе шаблона
5. Создание репозитория курса на основе шаблона
6. Настройка каталога курса
7. Выполнение заданий для самостоятельной работы # 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.

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 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.

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.

Благодаря тому, что Git является распределённой системой контроля версий,

резервную копию локального хранилища можно сделать простым копированием или архивацией.

Основные команды и их описание:

`git commit -am 'Описание коммита'`	сохранить все добавленные изменения и все изменённые файлы
`git checkout -b имя_ветки`	создание новой ветки, базирующейся на текущей
`git checkout имя_ветки`	переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
`git push origin имя_ветки`	отправка изменений конкретной ветки в центральный репозиторий
`git merge --no-ff имя_ветки`	слияние ветки с текущим деревом
`git branch -d имя_ветки`	удаление локальной уже слитой с основным деревом ветки
`git branch -D имя_ветки`	принудительное удаление локальной ветки
`git push origin :имя_ветки`	удаление ветки с центрального репозитория

Таблица 1

3 Выполнение лабораторной работы**

3.1 Настройка GitHub

Учётная запись GitHub у меня уже была, поэтому мне надо было просто зайти на сайт организации и авторизироваться. (Рис.1)

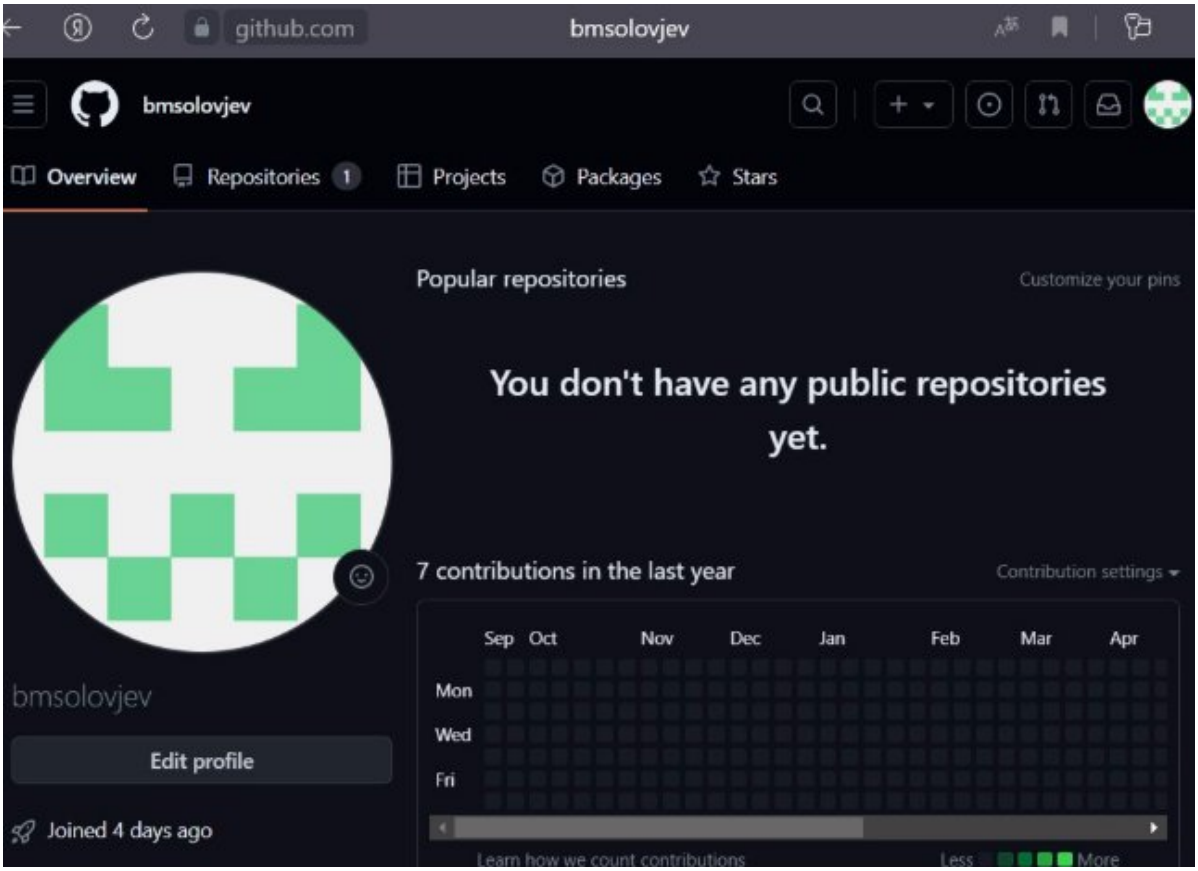

Рис.1 Аккаунт GitHub

3.2 Базовая настройка GitHub

Для того, чтобы выполнить предварительную конфигурацию Git, в своей виртуальной машине открываю терминал и ввожу команду “git config –global user.name ‘’ ”, а потом команду „git config –global user.email ‘<bogdan034@outlook.com>’” (рис.2).

```
[bmsolovjev@fedora ~]$ git config --global user.name "<bmsolovjev>"
[bmsolovjev@fedora ~]$ git config --global user.email "<bogdan034@outlook.com>"
```

Рис.2 Предварительная конфигурация Git

Настраиваю utf-8 в выводе сообщений (рис.3). [bmsolovjev@fedora ~]$ git config --glob

Рис.3 Настройка кодировки utf-8

Задаю имя «master» для начальной ветки (рис.4). [bmsolovjev@fedora ~]$ git config --gl

Рис.4 Создание имени «master» для начальной ветки Задаю параметр autocrlf (рис.5). [bmsolovjev@fedora ~]$ git config --global core.autocrlf input

Рис.5 Задавание параметра autocrlf Задаю параметр safecrlf (рис.6). [bmsolovjev@fedora ~]$ git config --global core.safecrlf warn

Рис.6 Задавание параметра safecrlf

3.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого использую команду «ssh-keygen -C “Bogdan Solovjev <bogdan034@outlook.com>”». Коюч автоматически сохраняется в каталоге ~/.ssh/.

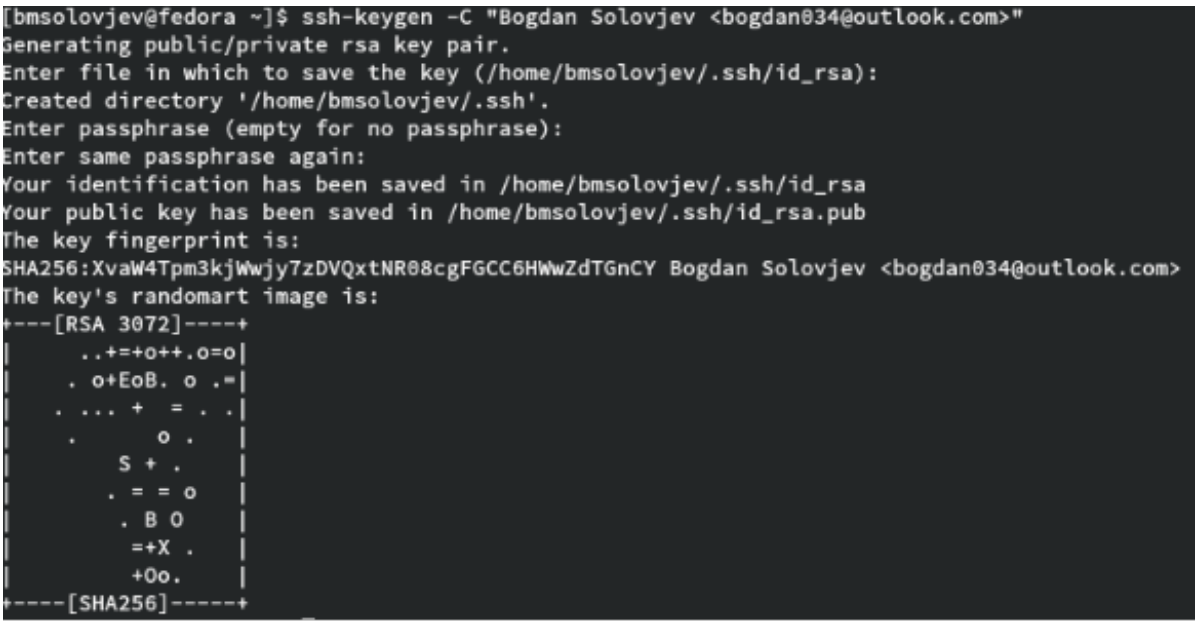

Рис.7 Создание SSH-ключа Копирую открытый ключ из директории ~/.ssh/

```
[bmsolovjev@fedora ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис.8 Копирование открытого ключа Вставляю этот ключ в специальное поле «Key»

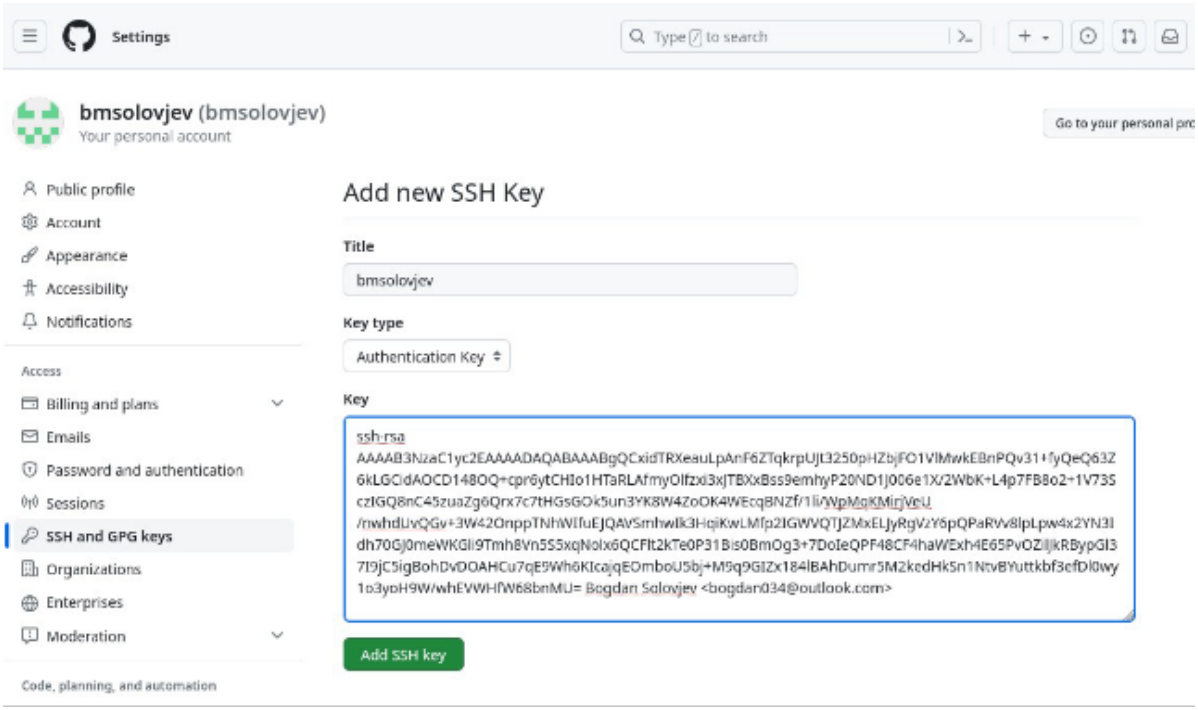

Рис.9 Добавление открытого ключа в GitHub

3.4 Создание рабочего пространства и репозитория курса на основе шаблона

С помощью команды «mkdir -p ~/work/study/2023-2024/“Архитектура компьютера”» создаю структурированное рабочее пространство (рис.10).

Рис.10 Создание директорий

3.5 Создание репозитория на основе шаблона

Перехожу по ссылке с шаблоном репозитория https://github.com/yamadharma/course-directory-student-template, создаю репезиторий и назваю его как на картинке (рис.11).

Рис.11 Создание репозитория по шаблону

Перехожу каталог курса с помощью команды «cd ~/work/study/2023- 2024/”Архитектура компьютера”». Потом клонирую созданный репозиторий командой «git clone –recursive gut@hub.com:bmsolovjev/study_2023-2024_arch-pc.git» (рис.12)

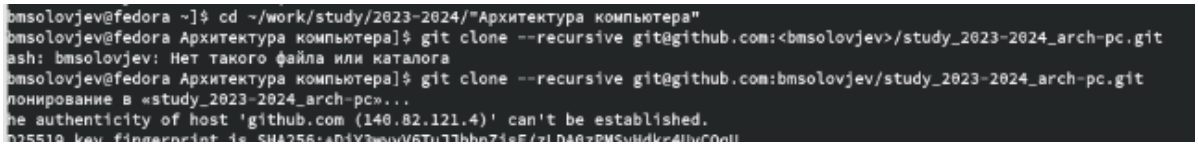

Рис.12 Переход в каталог и клонирование каталога

Ссылку для клонирования можно скопировать на странице созданного репозитория Code -> SSH:

3.6 Настройка каталога курса

Необходимо удалить файл package.json, для этого я перехожу в директорию study_2023-2024_arch-pc и использую команду «rm packege.json» (рис.13)

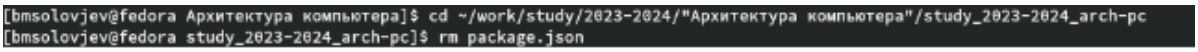

Рис.13 Переход в каталог и удаление файла Создаю необходимый каталог (рис.14)

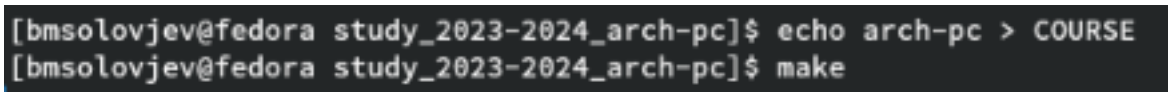

Рис.14 Создание каталога

Отправляю данные на сервер командой «git add», потом сохраняю изменения командой «git commit -am» (рис.15)

```
[bmsolovjev@fedora study_2023-2024_arch-pc]$ git add .
[bmsolovjev@fedora study_2023-2024_arch-pc]$ git commit -am 'feat(main): make course structure'
[master 0cba8dc] feat(main): make course structure
 199 files changed, 54725 insertions(+), 14 deletions(-)
 create mode 100644 labs/README.md
 create mode 100644 labs/README.ru.md
 create mode 100644 labs/lab01/presentation/Makefile
 create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
 create mode 100644 labs/lab01/presentation/presentation.md
 create mode 100644 labs/lab01/report/Makefile
 create mode 100644 labs/lab01/report/bib/cite.bib
 create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
 create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
 create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/__init__.py
```

Рис.15 Отправление и сохранение данных на сервере GitHub

Отправление изменений в центральный репозиторий командой «git push»(рис.16)

```
[bmsolovjev@fedora study_2023-2024_arch-pc]$ git push
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 342.13 КиБ | 2.50 МиБ/с, готово.
Всего 35 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:bmsolovjev/study_2023-2024_arch-pc.git
   13e62d0..0cba8dc  master -> master
```

Рис.16

4 Выполнение заданий для самостоятельной работы

1. Создаю в директории labs/lab02/report файл с отчётом по второй лабораторной работе.

```
[bmsolovjev@fedora study_2023-2024_arch-pc]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/study_2023-2024_arch-pc/labs/lab02/report
[bmsolovjev@fedora report]$ touch Л03_Соловьев_отчет
```

Рис.17 Создание отчёта

2. Так как я делаю отчёты не на виртуальной машине, мне будет проще вручную скопировать отчёты по лабораторным работам в каталог GitHub. (рис.18)

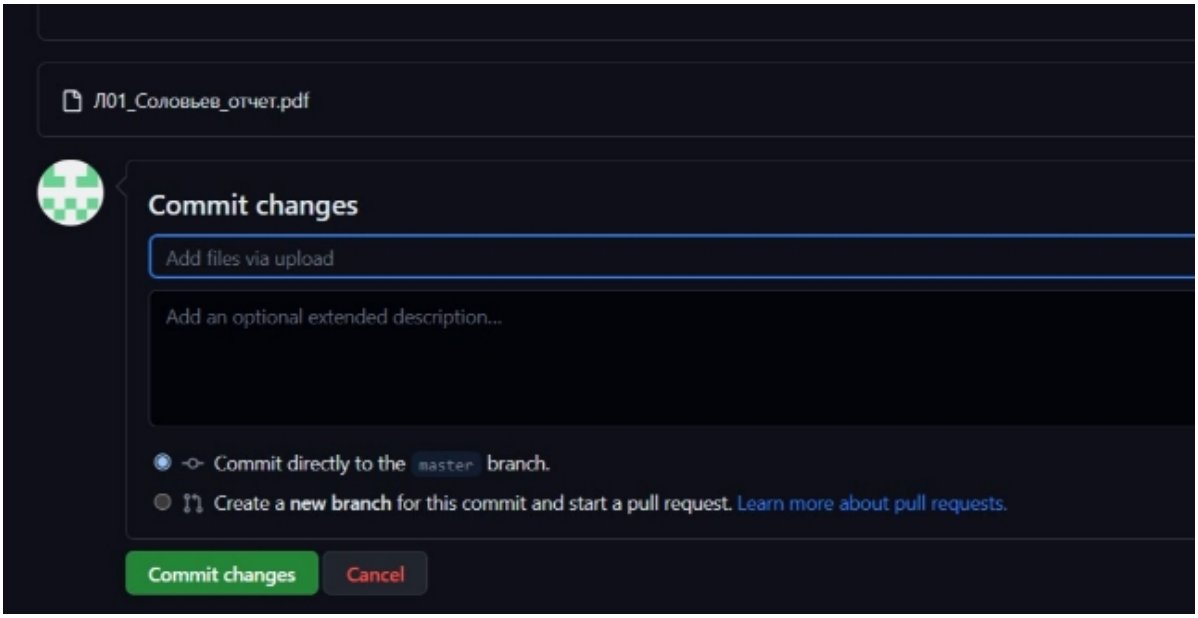

Рис.18 Перенос файла в каталог вручную

3. Загрузка файла второй лабораторной работы на Github (рис.19)

```
bmsolovjev@fedora report]$ touch Л02_Соловьев_отчет.pdf
bmsolovjev@fedora report]$ git add Л02_Соловьев_отчет.pdf
```

Рис.19 Создание и отправка файла

Сохраняю изменения командой «git commit -m “Add files”» (рис.20)

```
[bmsolovjev@fedora report]$ git commit -m "Add files"
[master dad8486] Add files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 labs/lab02/report/Л02_Соловьев_отчет
 create mode 100644 labs/lab02/report/Л02_Соловьев_отчет.pdf
[bmsolovjev@fedora report]$ git push -f origin master
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (5/5), 629 байтов | 314.00 КиБ/с, готово.
Всего 5 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:bmsolovjev/study_2023-2024_arch-pc.git
   0cba8dc..dad8486  master -> master
```

Рис.20 Сохранение изменений

5 Выводы

Выполнив данную работу, я изучил идеологию и применение средств контроля версий GitHub, научился взаимодействовать с сервером GitHub и создавать собственные репозитории, добавлять и копировать в них файлы, работать с шаблонами и создавать SSH-ключи.

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.

    1. 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 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд.

1. БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. 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.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с.

1. (Классика Computer Science).

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