Лабораторная работа №5

Дисциплина: Основы информационной безопасности

Феоктистов Владислав Сергеевич

Содержание

# 1 Цель работы

Целью данной работы является:

* изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов;
* получение практических навыков работы в консоли с дополнительными атрибутами;
* рассмотрение работы механизма смены идентификаторов процессов пользователей;
* изучение влияния Sticky-бита на запись и удаление файлов.

# 2 Задание

Первая часть задания:

* Написать программу simpleid.c, скомпилировать ее и сравнить результат ее выполнения с командой id;
* Усложнить программу, назвав его simpleid2.c и добавив вывод действительных идентификаторов и повторить предыдущие действия;
* От имени суперпользователя изменить владельца исполняемого файла simpleid2 и установить для нового владельца файла SetUID;
* Запустить исполняемый файл simpleid2 и снова сравнить результат с командой id;
* Повторить два предыдущих действия, установив SetGID;
* Создать программу readfile.c для чтения содержимого файлов и скомпилировать ее;
* Попробовать прочитать файл readfile.c, заранее отобрав права на чтение для пользователя guest и сменив владельца файла;
* Попробовать прочитать файл readfile.c и /etc/shadow через исполняемый файл readfile, заранее изменив владельца испольняемого файла и установив для него SetUID.

Вторая часть задания:

* Просмотреть наличие Stick-бита у каталога /tmp и попробовать создать, прочить и изменить текстовый файл в нем, а затем попытаться его удалить;
* Попробовать сделать тоже самое, только сняв атрибут t;
* Вернуть обратно Stick-бит для каталога /tmp.

# 3 Теоретическое введение

## 3.1 Изменение атрибутов

В ОС Linux права доступа к файлам, атрибуты и владение управляют уровнем доступа, который система обрабатывает, а пользователи имеют к файлам. Это гарантирует, что только авторизованные пользователи и процессы могут получить доступ к опеределенным файлам и каталогам. Атрибуты сосотят из девяти битов, которые и определяют права для разных групп пользователей. Первая тройка битов определяет права доступа для владельца, вторая тройка - для членов группы, последняя тройка - для всех остальных пользователей в системе. Каждая тройка битов (класс пользователей) определяет права на чтение, запись и исполнение. Эта концепция позволяет контролировать, какие пользователи могут читать, записывать (изменять) или выполнять файлы/каталоги.

Чтобы просмотреть права доступа к файлу, использовуется команда ls с опцией -l. Первый символ указывает тип файла. Это может быть обычный файл (-), каталог (d), символическая ссылка (l) или другие спицефические типы файлов. Следующие девять символов предоставляют доступ к файлу, три тройки по три символа каждая (три пользователя, три типа прав: r - чтение, w - запись, x - исполнение).

Права доступа к файлу/каталогу можно изменить с помощью команды chmod. Только root, владелец файла или пользователь с привилегией sudo могут изменять права доступа к файлу или каталогу. Разрешения можно указывать с помощью символьного, числового или справочного режимов [1].

## 3.2 Использование SetUID, SetGID и Sticky-bit для расширенной настройки прав досутпа в ОС Linux

**Setuid** – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволяет нам поднять привилегии пользователя в случае, если это необходимо. Классический пример использования этого бита в операционной системе это команда sudo.

Принцип работы **Setgid** очень похож на setuid с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

Последний специальный бит разрешения – это **Sticky Bit** . В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем.

Дополнительную информацию можно получить на сайте [2].

## 3.3 Реальные и эффективные идентификаторы

**Реальный идентификатор** - идентификатор пользователя, запустившего процесс. Эффективный идентификатор служит для определения прав доступа процесса к системным ресурсам (в первую очередь к ресурсам файловой системы).

**Эффективный идентификатор (effective id, effective gid)** - идентификатор пользователя или группы, получаемый процессом после вызова выполняемого файла; определяет права процесса. . Возможность изменения эффективных идентификаторов процесса удобна для организации абстрактных типов данных. Используя этот механизм, можно строить файлы, с которыми разрешено выполнять только определенный набор операций.

Обычно реальный и эффективный идентификаторы эквивалентны, т.е. процесс имеет в системе те же права, что и пользователь, запустивший его.

Дополнительную информацию можно получить на сайтах [3], [4].

## 3.4 Таблицы

Таблица 1: Описание некоторых каталогов файловой системы GNU Linux

| Имя каталога | Описание каталога |
| --- | --- |
| / | Корневая директория, содержащая всю файловую |
| /bin | Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям |
| /etc | Общесистемные конфигурационные файлы и файлы конфигурации установленных программ |
| /home | Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя |
| /media | Точки монтирования для сменных носителей |
| /root | Домашняя директория пользователя root |
| /tmp | Временные файлы |
| /usr | Вторичная иерархия для данных пользователя |

Таблица 2: Описание некоторых используемых в работе команд

| Команда | Описание команды |
| --- | --- |
| cat | Вывод содержимого указанного файла. |
| ls | Выводит содержимое каталога. Опция -l выводит дополнительную информацию, -a отображает скрытые файлы, в названии которых в самом начале стоит символ ‘.’ |
| id | Выводи UID (идентификатор пользователя), GID (идентификатор группы пользователя), groups (основные группы пользователя) |
| chmod | Изменение прав доступа к файлам и каталогам, используемых в Unix-подобных операционных системах. |
| chown | Изменение владельца и группы пользователей файла или каталога, используемых в Unix-подобных операционных системах. |
| echo | Вывод переданных аргументов, строки, текста. |
| touch | Создает текстовый файл по указанному пути и с указанным именем внутри пути. |
| rm | Удаляет файл(ы) (каталог(и) при указании опции -r) по указанному(ым) пути(ям). |
| cd | Перемещение по файловой системе. |
| grep | Дает возможность вести поиск строкт. Также можно передать вывод любой команды в grep, что сильно упрощает работу во время поиска |
| whereis | Выводит расположение бинарных или исходных файлов. |
| gcc | Копиляция исходного C файла в исполняемый (объектный) файл. |

Более подробно об Unix см. в [5–10].

# 4 Выполнение лабораторной работы

## 4.1 Подготовка лабораторного стенда

Перед выполнением лабораторной работы стоит убедиться в том, что gcc уже предустановлен в системе, для этого введем команду *gcc -v*. Отсутствие сообщения об ошибки и наличие о версии gcc, говорит о том, что gcc уже установлен в системе. Кроме того, так как программы с установленным битом SetUID могут представлять большую брешь в системе безопасности, в современных системах используются дополнитнльные механизмы защиты. Чтобы защита системы не мешала выполнению работы, необходимо будет ее отключить, для этого нужно будет ввести команду *setenforce 0* от имени суперпользователя. В случае успеха команда *getenforce* должна вывести сообщение *Permissive*.

Дополнительно можно посмотреть пути до компиляторов для языков C и C++ [**cmds:** *whereis gcc* и *whereis g++*] (рис. 1).

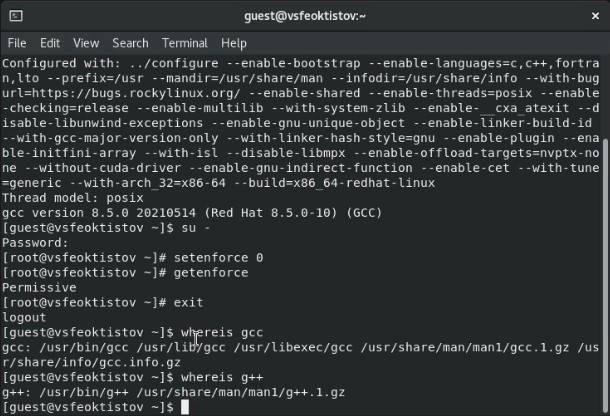


Рис. 1: Подготовка лабораторного стенда

## 4.2 Создание программы

Для начала войдем в систему под пользователем guest после запуска вирутральной машины и операционной системы, а после создадим программу *simpleid.c* (рис. 2), как это указано в документе с описанием выполнения лабораторной работы №5 [**cmds:** *touch simpleid.c* и *nano simpleid.c*]. Для сохранения изменений нужно сажать клавиши Ctrl + X, ввести y и нажать Enter. Провить, что исходный код успешно сохранился, можно командой cat [**cmd:** *cat simpleid.c*].

После компиляции [**cmd:** *gcc simpleid.c -o simpleid*] и запуска исполняемого файла [**cmd:** *./simpleid*] сравним результат работы программы с выводом команды *id*. Как можно заметить идентификторы uid и gid совпадают в обоих случаях (рис. 3).



Рис. 2: Написание кода simpleid.c

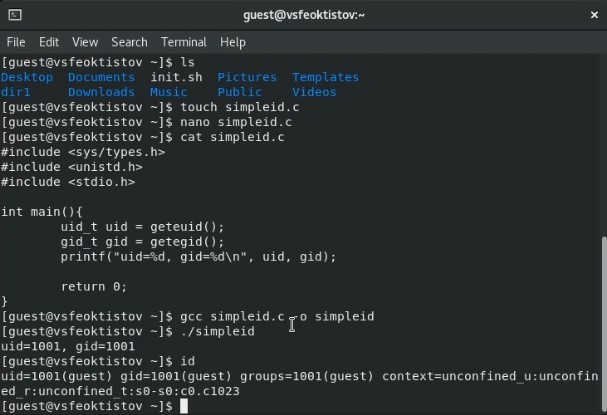


Рис. 3: Создание файла simpleid.c и запуск программы

Усложним программу, добавив вывод действительных идентификаторов (рис. 4). Программу сохраним под названием *simpleid2.c*. Для удобства проще будет скопировать файл *simpleid.c* в этот же каталог, но уже под названием *simpleid2.c* [**cmd:** *cp simpleid.c simpleid2.c*], а затем уже отредактировать его.

После компиляции [**cmd:** *gcc simpleid2.c -o simpleid2*] и запуска исполняемого файла [**cmd:** *./simpleid2*] сравним результат работы программы с выводом команды *id*. Как можно заметить реальные, эффективные и пользовательские идентификторы совпадают (рис. 5).

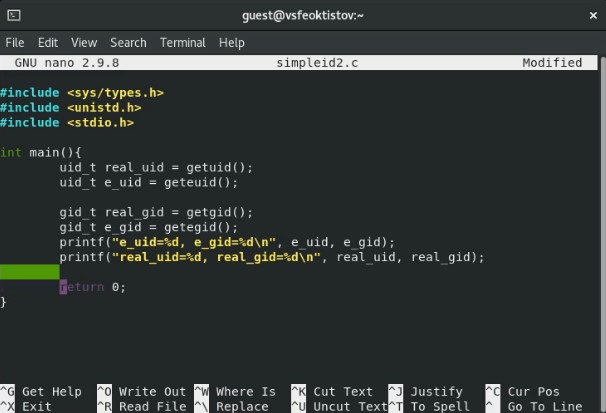


Рис. 4: Написание кода simpleid2.c

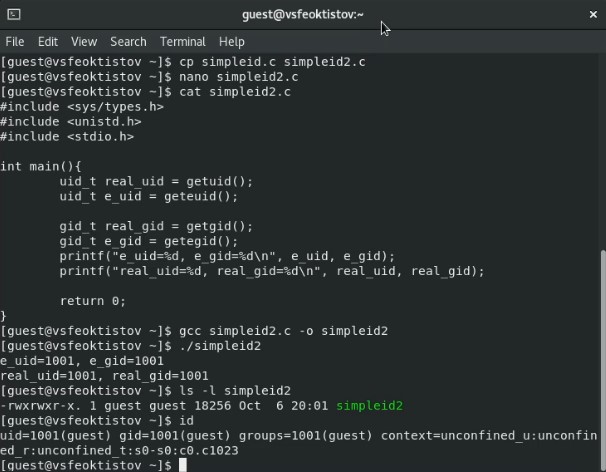


Рис. 5: Создание файла simpleid2.c и запуск программы

От имени суперпользователя [**cmd:** *su -*] изменим владельца исполняемого файла *simpleid2* [**cmd:** *chown root:guest /home/guest/simpleid2*], установим для нового владельца дополнительный атрибут s (SetUID) [**cmd:** *chmod u+s /home/guest/simpleid2*] и проверим как изменились атрибуты прав и владельцы файла [**cmd:** *ls -l /home/guest/simpleid2*]. Как видно в первой тройке битов вместо привычного символа x стоит символ s, что и говорит об успешной установке SetUID. После выполнения этих команд нужно вернуться к пользователю guest [**cmd:** *exit*] (рис. 6).

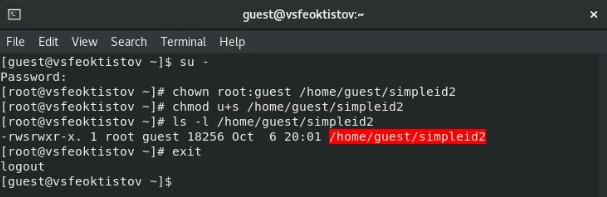


Рис. 6: Смена владельца файла и добавление атрибута s

Повторно запустим исполняемый файл после вышеописанных действий и сравним результат выполнения программы с выводом команды *id*. Как можно заметить, теперь e\_uid отличается от real\_uid и uid в id. Это произошло, поскольку файл запускался от имени пользователя guest, а исполнялся от пользователя root (у root-пользователя uid=0) (рис. 7).

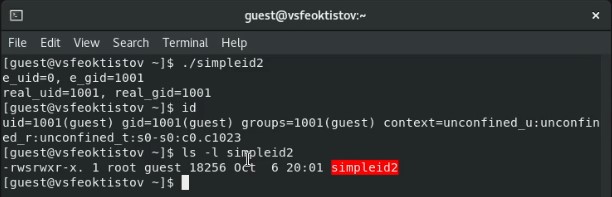


Рис. 7: Повторный запуск simpleid2 с установленным SetUID-битом

Проделаем те же самые действия только с еще установленным SetGID-битом (рис. 8).

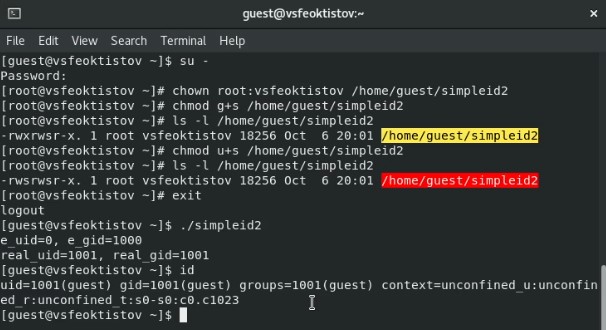


Рис. 8: Повторный запуск simpleid2 с установленным SetGID-битом

В результате, теперь еще и e\_gid отличается от real\_gid и gid пользователя. Теперь программа исполняется от новой уствновленной группы (*vsfeoktistov* gid=1000) (рис. 8).

Далее создадим программу readfile.c, которая будет читать содержимое указанного файла (рис. 9 и 10).

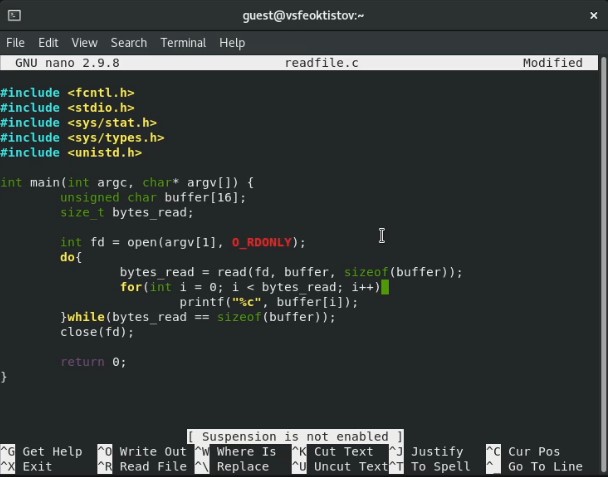


Рис. 9: Написание кода readfile.c

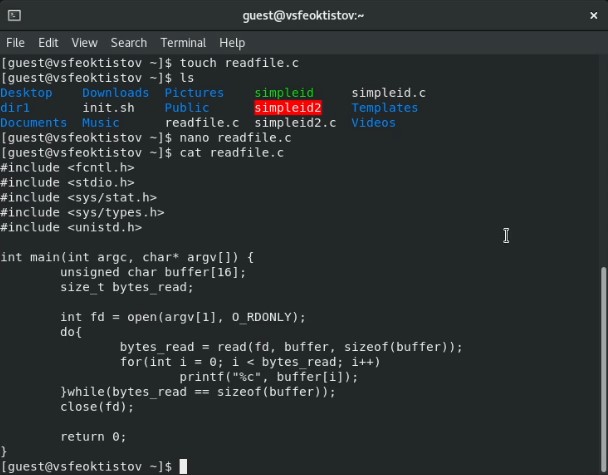


Рис. 10: Создание файла readfile.c

Скомпилируем файл [**cmd:** *gcc readfile.c -o readfile*], сменим владельца файла *readfile.c* и изменим права так, чтобы только суперпользователь (*root*) мог прочитать его, а guest не мог [**cmds:** *chown root:guest /home/guest/readfile.c* и *chmod 700 /home/guest/readfile.c*]. Как можно видеть, теперь пользователь *guest* не может его прочитать (рис. 11).

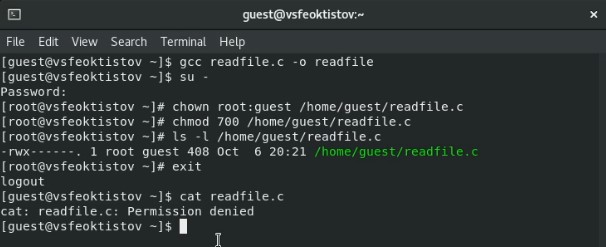


Рис. 11: Компиляция и изменение прав и владельца файла readfile.c

Далее сменим владельца программы *readfile* и установим SetUID-бит [**cmds:** *chown root:guest /home/guest/readfile* и *chmod u+s /home/guest/readfile*]. Затем проверим: может ли программа *readfile* прочитать файл *readfile.c* [**cmd:** *./readfile readfile.c*]. Как видно из рисунка 12, программа успешно прочитала содержимое, поскольку имполнилась от имени владельца, который имеет право чтения файла *readfile.c*. Кроме того, программа также может вывести содержимое файла */etc/shadow* [**cmd:** *./readfile /etc/shadow*] (рис. 12).

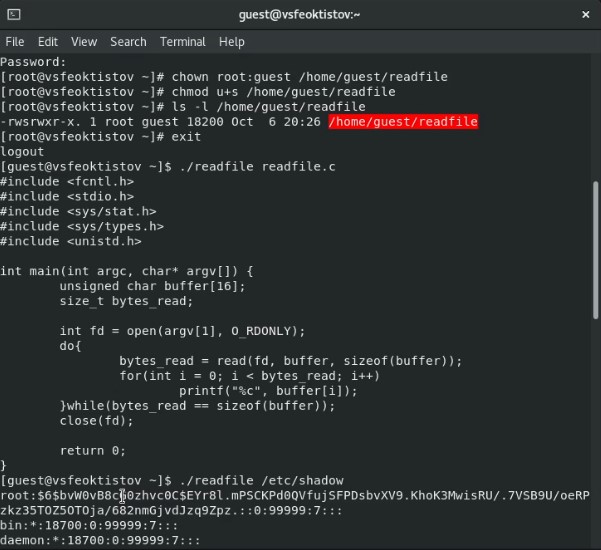


Рис. 12: Смена владельца и атрибутов с проверкой работы программы readfile

## 4.3 Исследование Sticky-бита

В первую очередь, выясним, установлен ли атрибут Sticky на директорию */tmp* [**cmd:** *ls -l / | grep tmp*]. Символ *t* вместо *x* в конце, в третьей тройке битов атрибутов, указывает на его наличие. От имени пользователя *guest* создадим файл *file01.txt* в директории */tmp* со словом *test* [**cmd:** *echo “test” > /tmp/file01.txt*], а после просмотрим атрибуты у только что созданного файла [**cmd:** *ls -l /tmp/file01.txt*] и разрешим чтение и запись жля категории пользователей “все остальные” [**cmd:** *chmod o+rw /tmp/file01.txt*] (рис. 13).

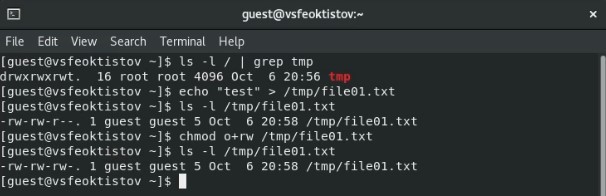


Рис. 13: Проверка Stick-бита и создание файла с определенными правами

Далее от имени пользователя *guest2* [**cmd:** *su - guest2*] попробуем прочитать, дозаписать, перезаписать и удалить файл *file01.txt*. Как видно из рисунка 14, все работает, кроме удаления, поскольку Sticky-бит разрешает удаление файлов в каталоге только владельцу этого каталога.

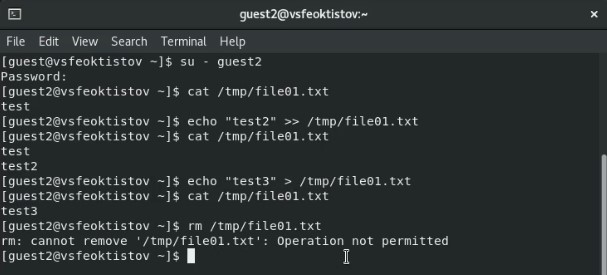


Рис. 14: Работа с Sticky-битом

Повысим права до суперпользователя [**cmd:** *su -*], выполним команду, снимающую атрибут t (Sticky-бит) с директории */tmp* [**cmd:** *chmod -t /tmp*] и покинем режим суперпользователя [**cmd:** *exit*] (рис. 15).

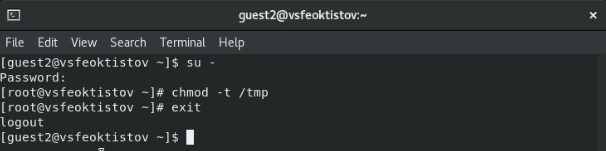


Рис. 15: Снятие Sticky-бита с каталога /tmp

Повторим предыдущие шаги и пронаблюдаем изменения. В результате, все команды сработали, в том числе и удаление файла из директории (рис. 16).

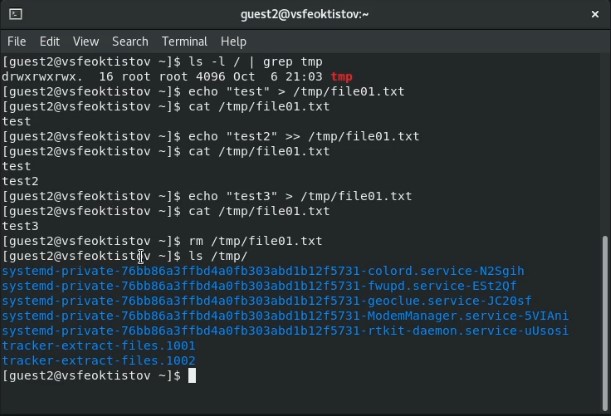


Рис. 16: Работа без Sticky-бита

После окончания работы необходимо вернуть атрибут t на директорию */tmp*, чтобы сохранить безопасность и функциональность системы (рис. 17).

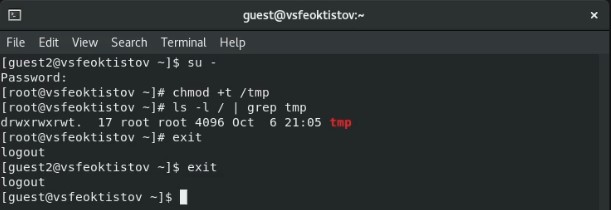


Рис. 17: Возвращение Sticky-бита

# 5 Выводы

В процессе выполнения лабораторной работы:

* изученил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов;
* получил практические навыки работы в консоли с дополнительными атрибутами;
* рассмотрел работу механизма смены идентификаторов процессов пользователей;
* изученил влияние Sticky-бита на запись и удаление файлов.

# Список литературы

1. Понимание прав доступа к файлам в Linux [Электронный ресурс]. Baks, 2021. URL: <https://baks.dev/article/terminal/understanding-linux-file-permissions?ysclid=l8czjs1hnp553393513>.

2. Использование SETUID, SETGID и Sticky bit для расширенной настройки прав доступа в операционных системах Linux [Электронный ресурс]. RuVDS, 2021. URL: <https://ruvds.com/ru/helpcenter/suid-sgid-sticky-bit-linux/>.

3. Эффективный идентификатор [Электронный ресурс]. Большая энциклопедия нефти и газа, -. URL: <https://www.ngpedia.ru/id43832p1.html>.

4. Реальный (RID) и эффективный (EUID) идентификаторы пользователя [Электронный ресурс]. В помощь веб-мастеру, -. URL: <https://wm-help.net/lib/b/book/173895509/64>.

5. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.

6. Newham C. [Learning the bash Shell: Unix Shell Programming](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658). O’Reilly Media, 2005. 354 с.

7. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

8. Robbins A. [Bash Pocket Reference](https://www.ncbi.nlm.nih.gov/pubmed/25246403). O’Reilly Media, 2016. 156 с.

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

10. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.