

# **Лабораторная работа №5**

**Информационная безопасность**

Николаев Дмитрий Иванович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
2.1	SetUID . . . . .	6
2.2	Sticky-бит . . . . .	6
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Создание программы . . . . .	7
3.2	Исследование Sitcky-бита . . . . .	13
<b>4</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

## Список иллюстраций

3.1	Проверка установленности gcc . . . . .	7
3.2	Создание файла simpleid.c от имени guest . . . . .	8
3.3	Код файла simpleid.c . . . . .	8
3.4	Компиляция и запуск программы simpleid . . . . .	8
3.5	Код программы simpleid2.c . . . . .	9
3.6	Компиляция и исполнение файла simpleid2 . . . . .	9
3.7	Изменение владельца и прав доступа к файлу simpleid2 . . . . .	9
3.8	Проверка атрибутов файла simpleid2 и вывод для root пользователя	10
3.9	Вывод simpleid2 для пользователя guest . . . . .	10
3.10	Установка SetGID-бита и проверка вывода файла simpleid2 . . . .	10
3.11	Код программы readfile.c . . . . .	11
3.12	Смена владельца и прав на файл readfile.c с проверкой . . . . .	11
3.13	Смена владельца и установление SetUID-бита на файл readfile . .	12
3.14	Чтение программы readfile.c с помощью readfile . . . . .	12
3.15	Чтение программы/etc/shadow с помощью readfile . . . . .	13
3.16	Проверка Sticky-бита и создание файла file01.txt с правами на чтение и запись . . . . .	14
3.17	Установка прав на запись и чтение file01.txt . . . . .	14
3.18	Проверка некоторых действий с файлом file01.txt от имени пользователя guest2 . . . . .	15
3.19	Проверка некоторых действий с файлом file01.txt без Sticky-бита от имени пользователя guest2 . . . . .	16
3.20	Возвращение Sticky-бита на директории /tmp . . . . .	16
3.21	Проверка атрибутов на директории /tmp . . . . .	17

## **Список таблиц**

# 1 Цель работы

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в консоли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

## 2 Теоретическое введение

### 2.1 SetUID

SetUID (сокращения от англ. set user ID upon execution — “установка ID пользователя во время выполнения”) разрешает пользователям запускать исполняемые файлы с правами владельца исполняемого файла. Иногда файлы требуют разрешения на выполнение для пользователей, которые не являются членами группы владельца, в этом случае потребуется предоставить специальные разрешения на выполнение. Когда SetUID установлен, пользователь может запускать любую программу, как её владелец.

### 2.2 Sticky-бит

В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем. Пример использования этого бита в операционной системе это системная папка /tmp . Эта папка разрешена на запись любому пользователю, но удалять файлы в ней могут только пользователи, являющиеся владельцами этих файлов. Символ «t» указывает, что на папку установлен Sticky-бит.

## 3 Выполнение лабораторной работы

С помощью команды `gcc -v` убедимся, что у меня установлен компилятор `gcc` ([3.1]).

```
[dinikolaev@dinikolaev ~]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-host-pie --enable-host-bind-now --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-shared --enable-threads=posix --enable-checking=release --with-system-zlib --enable-_cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-plugin --enable-initfini-array --without-isl --enable-multilib --with-linker-hash-style=gnu --enable-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tune=generic --with-arch_64=x86-64-v2 --with-arch_32=x86-64 --build=x86_64-redhat-linux --with-build-config=bootstrap-lto --enable-link-serialization=1
Модель многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)
[dinikolaev@dinikolaev ~]$
```

Рис. 3.1: Проверка установленности `gcc`

### 3.1 Создание программы

1. Действуя согласно [1], зайдём от имени пользователя `guest` и создадим файл `simpleid.c` ([3.2]–[3.3]).

```
[dinikolaev@dinikolaev ~]$ su - guest
Пароль:
[guest@dinikolaev ~]$ ls
dir1  file0  Документы  Изображения  Общедоступные  Шаблоны
dir2  Видео  Загрузки  Музыка  'Рабочий стол'
[guest@dinikolaev ~]$ touch simpleid.c
[guest@dinikolaev ~]$ ls
dir1  file0  Видео  Загрузки  Музыка  'Рабочий стол'
dir2  simpleid.c  Документы  Изображения  Общедоступные  Шаблоны
[guest@dinikolaev ~]$
```

Рис. 3.2: Создание файла simpleid.c от имени guest

```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int
5 main ()
6 {
7     uid_t uid = geteuid ();
8     gid_t gid = getegid ();
9     printf ("uid=%d, gid=%d\n", uid, gid);
10    return 0;
11 }
```

Рис. 3.3: Код файла simpleid.c

2. Скомпилируем программу и убедимся, что исполняемый файл программы создан: `gcc simpleid.c -o simpleid`. Выполним программу `simpleid`: `./simpleid` и сравним с выводом системной программы `id`. Как можно заметить, выходы идентичны ([3.4]).

```
[guest@dinikolaev ~]$ gcc simpleid.c -o simpleid
[guest@dinikolaev ~]$ ./simpleid
uid=1001, gid=1001
[guest@dinikolaev ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@dinikolaev ~]$
```

Рис. 3.4: Компиляция и запуск программы simpleid

3. Усложним программу, добавив вывод действительных идентификаторов. Получившуюся программу назовём `simpleid2.c` ([3.5]). После скомпилируем и запустим получившийся исполняемый файл ([3.6]).



```

1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int
5 main ()
6 {
7     uid_t real_uid = getuid ();
8     uid_t e_uid = geteuid ();
9     gid_t real_gid = getgid ();
10    gid_t e_gid = getegid ();
11    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
12    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
13    return 0;
14 }

```

Рис. 3.5: Код программы simpleid2.c

```

[guest@dinikolaev ~]$ touch simpleid2.c
[guest@dinikolaev ~]$ gcc simpleid2.c -o simpleid2
[guest@dinikolaev ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@dinikolaev ~]$

```

Рис. 3.6: Компиляция и исполнение файла simpleid2

4. От имени суперпользователя выполним команды ([3.7]):

- `chown root:guest /home/guest/simpleid2`
- `chmod u+s /home/guest/simpleid2`

```

[guest@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest]# chown root:guest /home/guest/simpleid2
[root@dinikolaev guest]# chmod u+s /home/guest/simpleid2

```

Рис. 3.7: Изменение владельца и прав доступа к файлу simpleid2

Команда `chown root:guest /home/guest/simpleid2` меняет владельца файла (с `guest` на `root`). Команда `chmod u+s /home/guest/simpleid2` меняет права доступа к файлу (добавляет атрибут `s` (вместо `x` у владельца файла)).

5. Выполним проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`: `ls -l simpleid2` и запустим `simpleid2` и `id`. Для пользователя `root` все выводы равняются 0 ([3.8]).

```
[root@dinikolaev guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 24488 окт  3 15:39 simpleid2
[root@dinikolaev guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@dinikolaev guest]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 3.8: Проверка атрибутов файла `simpleid2` и вывод для `root` пользователя

6. Проверим вывод для пользователя `guest` ([3.9]).

```
[root@dinikolaev guest]# su - guest
[guest@dinikolaev ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@dinikolaev ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@dinikolaev ~]$
```

Рис. 3.9: Вывод `simpleid2` для пользователя `guest`

7. Прделаем тоже самое относительно `SetGID`-бита, предварительно сняв `UID`-бит ([3.10]).

```
[guest@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest]# chmod u-s /home/guest/simpleid2
[root@dinikolaev guest]# chmod g+s /home/guest/simpleid2
[root@dinikolaev guest]# ls -l simpleid2
-rwxr-sr-x. 1 root guest 24488 окт  3 15:39 simpleid2
[root@dinikolaev guest]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@dinikolaev guest]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@dinikolaev guest]# su - guest
[guest@dinikolaev ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@dinikolaev ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@dinikolaev ~]$ ls -l simpleid2
-rwxr-sr-x. 1 root guest 24488 окт  3 15:39 simpleid2
[guest@dinikolaev ~]$
```

Рис. 3.10: Установка `SetGID`-бита и проверка вывода файла `simpleid2`

8. Создадим программу readfile.c ([3.11]) и скомпилируем её: gcc readfile.c -o readfile. Сменим владельца у файла readfile.c и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог, совершив проверку выполненного ([3.12]).

```
1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <sys/stat.h>
4 #include <sys/types.h>
5 #include <unistd.h>
6 int
7 main (int argc, char* argv[])
8 {
9     unsigned char buffer[16];
10    size_t bytes_read;
11    int i;
12    int fd = open (argv[1], O_RDONLY);
13    do
14    {
15        bytes_read = read(fd, buffer, sizeof(buffer));
16        for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
17    } while (bytes_read == sizeof (buffer));
18    close (fd);
19    return 0;
20 }
```

Рис. 3.11: Код программы readfile.c

```
[guest@dinikolaev ~]$ touch readfile.c
[guest@dinikolaev ~]$ gcc readfile.c -o readfile
[guest@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest]# chown root:guest /home/guest/readfile.c
[root@dinikolaev guest]# chmod 700 /home/guest/readfile.c
[root@dinikolaev guest]# ls -l readfile.c
-rwx-----. 1 root guest 414 окт  3 15:52 readfile.c
[root@dinikolaev guest]# su - guest
[guest@dinikolaev ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@dinikolaev ~]$
```

Рис. 3.12: Смена владельца и прав на файл readfile.c с проверкой

9. Сменим у программы readfile владельца и установим SetUID-бит ([3.13]).

```
[guest@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest]# chown root:guest /home/guest/readfile
[root@dinikolaev guest]# chmod u+s /home/guest/readfile
[root@dinikolaev guest]# ls -l readfile
-rwsr-xr-x. 1 root guest 24432 окт  3 15:52 readfile
[root@dinikolaev guest]# su - guest
```

Рис. 3.13: Смена владельца и установление SetUID-бита на файл readfile

10. Проверим, может ли программа readfile прочитать файл readfile.c ([3.14]) и /etc/shadow ([3.15]). Так как мы установили SetUID-бит программе readfile, то ей временно предоставляются права владельца файла.

```
[guest@dinikolaev ~]$ ./readfile readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;
    int fd = open (argv[1], O_RDONLY);
    do
    {
        bytes_read = read(fd, buffer, sizeof(buffer));
        for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
    } while (bytes_read == sizeof (buffer));
    close (fd);
    return 0;
}
```

Рис. 3.14: Чтение программы readfile.c с помощью readfile

```
[guest@dinikolaev ~]$ ./readfile /etc/shadow
root:$6$uzIWDVIM3DxZGIjE$9qg5cxA6E0TfuNBIgx2Z0//pL88Kd9wyiGtIFzRrpFV6mzYk7gvNvn
Ysr71YYz64WuASk0wdSMLWw3X133Fx1::0:99999:7:::
bin:!:19347:0:99999:7:::
daemon:!:19347:0:99999:7:::
adm:!:19347:0:99999:7:::
lp:!:19347:0:99999:7:::
sync:!:19347:0:99999:7:::
shutdown:!:19347:0:99999:7:::
halt:!:19347:0:99999:7:::
mail:!:19347:0:99999:7:::
operator:!:19347:0:99999:7:::
games:!:19347:0:99999:7:::
ftp:!:19347:0:99999:7:::
nobody:!:19347:0:99999:7:::
systemd-coredump:!:19609::::::
dbus:!:19609::::::
polkitd:!:19609::::::
avahi:!:19609::::::
rtkit:!:19609::::::
libstoragemgmt:!:19609::::::
systemd-oom:!:19609::::::
geoclue:!:19609::::::
tss:!:19609::::::
cockpit-ws:!:19609::::::
cockpit-wsinstance:!:19609::::::
colord:!:19609::::::
sssd:!:19609::::::
setroubleshoot:!:19609::::::
pipewire:!:19609::::::
flatpak:!:19609::::::
clevis:!:19609::::::
gdm:!:19609::::::
gnome-initial-setup:!:19609::::::
pesign:!:19609::::::
sshd:!:19609::::::
chrony:!:19609::::::
dnsmasq:!:19609::::::
tcpdump:!:19609::::::
dinikolaev:$6$0G3AyU5kR.xS1jM$M2E7uM4922DFie0dxezGtNytTvaK6tvwGK0JieU6XiTVo40HPA
i/CojkdE4j0Qo0ZswzQb8cnS2IHu9mDSSFCw1::0:99999:7:::
vboxadd:!:19609::::::
guest:$6$etabJ3/NOQ3ix/OV$H010bUdzNj14goI7Kw.RN.KQXXft2x7AtayNDbv33XA2.tnf3jPnEi
ez2f2DJwGC181NZTt1csS339fZYqt0:19613:0:99999:7:::
guest2:$6$Wxw4hbXdwKXoFwta$24cbNSXNtNRSkEegqlf8vhNcNh0hzoOP/Xh06Sfy52WR40UNI0oPV
mUQKwnccBP6F50o9FHfNwx7joom6Xvm5.:19623:0:99999:7:::
[guest@dinikolaev ~]$
```

Рис. 3.15: Чтение программы/etc/shadow с помощью readfile

## 3.2 Исследование Sitcky-бита

1. Выясним, установлен ли атрибут Sticky на директории /tmp, для чего выполним команду `ls -l / | grep tmp`. От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt`. Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» ([3.16]):

- `ls -l /tmp/file01.txt`

- `chmod o+rw /tmp/file01.txt`
- `ls -l /tmp/file01.txt`

```
[guest@dinikolaev ~]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 окт  3 15:57 tmp
[guest@dinikolaev ~]$ echo "test" > /tmp/file01.txt
[guest@dinikolaev ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 окт  3 16:02 /tmp/file01.txt
[guest@dinikolaev ~]$ chmod o+rw /tmp/file01.txt
[guest@dinikolaev ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 окт  3 16:02 /tmp/file01.txt
[guest@dinikolaev ~]$
```

Рис. 3.16: Проверка Sticky-бита и создание файла file01.txt с правами на чтение и запись

Установим права на запись и чтение для всех категорий пользователей ([3.17]).

```
[guest@dinikolaev ~]$ su - guest2
Пароль:
[guest2@dinikolaev ~]$ cat /tmp/file01.txt
test
[guest2@dinikolaev ~]$ echo "test2" > /tmp/file01.txt
-bash: /tmp/file01.txt: Отказано в доступе
[guest2@dinikolaev ~]$ echo "test2" >> /tmp/file01.txt
-bash: /tmp/file01.txt: Отказано в доступе
[guest2@dinikolaev ~]$ chmod 555 /tmp/file01.txt
chmod: изменение прав доступа для '/tmp/file01.txt': Операция не позволена
[guest2@dinikolaev ~]$ su - guest
Пароль:
[guest@dinikolaev ~]$ chmod 555 /tmp/file01.txt
[guest@dinikolaev ~]$ ls -l /tmp/file01.txt
-r-xr-xr-x. 1 guest guest 5 окт  3 16:02 /tmp/file01.txt
[guest@dinikolaev ~]$ chmod 666 /tmp/file01.txt
[guest@dinikolaev ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 окт  3 16:02 /tmp/file01.txt
[guest@dinikolaev ~]$
```

Рис. 3.17: Установка прав на запись и чтение file01.txt

- От имени пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt: `cat /tmp/file01.txt`, дозаписать текст в него командой `echo "test2" > /tmp/file01.txt`, перезаписать файл командой `echo "test3" > /tmp/file01.txt` и попробуем удалить файл. Все действия кроме удаления выполнить удалось ([3.18]).

```
[guest@dinikolaev ~]$ su - guest2
Пароль:
[guest2@dinikolaev ~]$ cat /tmp/file01.txt
test
[guest2@dinikolaev ~]$ echo "test2" >> /tmp/file01.txt
[guest2@dinikolaev ~]$ cat /tmp/file01.txt
test
test2
[guest2@dinikolaev ~]$ echo "test3" > /tmp/file01.txt
[guest2@dinikolaev ~]$ cat /tmp/file01.txt
test3
[guest2@dinikolaev ~]$ rm /tmp/file01.txt
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена
[guest2@dinikolaev ~]$
```

Рис. 3.18: Проверка некоторых действий с файлом file01.txt от имени пользователя guest2

3. Снимем атрибут t (Sticky-бит) с файла file01.txt от имени суперпользователя и повторим действия из предыдущего шага ([3.19]). Помимо уже успешно выполнимых действий с файлом, мы теперь смогли удалить файл от имени пользователя, не являющегося его владельцем. Таким образом, Sticky-bit позволяет защищать файлы от случайного удаления, когда несколько пользователей имеют права на запись в один и тот же файл. Если у файла атрибут t стоит, значит пользователь может удалить файл, только если он является пользователем-владельцем файла или каталога, в котором содержится файл. Если же этот атрибут не установлен, то удалить файл могут все пользователи, которым позволено удалять файлы из каталога.

```

[guest2@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest2]# chmod -t /tmp
[root@dinikolaev guest2]# exit
exit
[guest2@dinikolaev ~]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 окт  3 16:30 tmp
[guest2@dinikolaev ~]$ echo "test2" >> /tmp/file01.txt
[guest2@dinikolaev ~]$ cat /tmp/file01.txt
test3
test2
[guest2@dinikolaev ~]$ echo "test3" > /tmp/file01.txt
[guest2@dinikolaev ~]$ cat /tmp/file01.txt
test3
[guest2@dinikolaev ~]$ rm /tmp/file01.txt

```

Рис. 3.19: Проверка некоторых действий с файлом file01.txt без Sticky-бита от имени пользователя guest2

4. Повысим свои права до суперпользователя и вернём атрибут t на директорию /tmp ([3.20]) и проверим возвращение атрибута ([3.21]).

```

[guest2@dinikolaev ~]$ su -
Пароль:
[root@dinikolaev ~]# chmod +t \tmp
chmod: невозможно получить доступ к 'tmp': Нет такого файла или каталога
[root@dinikolaev ~]# chmod +t /tmp
[root@dinikolaev ~]# exit
выход

```

Рис. 3.20: Возвращение Sticky-бита на директории /tmp



```

[guest2@dinikolaev ~]$ ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 окт  3 16:34 tmp
[guest2@dinikolaev ~]$ ls /tmp
ls: невозможно получить доступ к 'tmp': Нет такого файла или каталога
[guest2@dinikolaev ~]$ ls /tmp
systemd-private-1a31459a150e471bb36cd3237fe865fb-chrond.service-dxHpb4
systemd-private-1a31459a150e471bb36cd3237fe865fb-colord.service-kIqvGN
systemd-private-1a31459a150e471bb36cd3237fe865fb-dbus-broker.service-ga2XIF
systemd-private-1a31459a150e471bb36cd3237fe865fb-fwupd.service-GW20lm
systemd-private-1a31459a150e471bb36cd3237fe865fb-ModemManager.service-sCb8dQ
systemd-private-1a31459a150e471bb36cd3237fe865fb-power-profiles-daemon.service-H
uNTmH
systemd-private-1a31459a150e471bb36cd3237fe865fb-rtkit-daemon.service-9SreMf
systemd-private-1a31459a150e471bb36cd3237fe865fb-switcheroo-control.service-tc1Z
dS
systemd-private-1a31459a150e471bb36cd3237fe865fb-systemd-logind.service-4T0iCC
systemd-private-1a31459a150e471bb36cd3237fe865fb-upower.service-iYHbcd
Temp-32c0ce52-029b-4d8f-9443-ee095c9d7493
tracker-extract-3-files.1000
vboxguest-Module.symvers
[guest2@dinikolaev ~]$

```

Рис. 3.21: Проверка атрибутов на директории /tmp

## 4 Выводы

В ходе выполнения лабораторной работы я изучил механизмы изменения идентификаторов, применения SetUID и Sticky-битов. Я получила практические навыки работы в консоли с дополнительными атрибутами. Я рассмотрела работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

## Список литературы

1. Кулябов Д. С., Королькова А. В., Геворкян М. Н Лабораторная работа №5 [Электронный ресурс]. RUDN, 2023. URL: [https://esystem.rudn.ru/pluginfile.php/2090208/mod\\_resource/content/2/005-lab\\_discret\\_sticky.pdf](https://esystem.rudn.ru/pluginfile.php/2090208/mod_resource/content/2/005-lab_discret_sticky.pdf).