

# **Лабораторная работа №5**

**Дисциплина: Информационная безопасность**

Губина Ольга Вячеславовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</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>
4.1	Создание программы . . . . .	9
4.2	Исследование Sticky-бита . . . . .	18
<b>5</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

## Список иллюстраций

4.1	Проверка необходимых ресурсов . . . . .	9
4.2	Создание и компиляция файла . . . . .	10
4.3	Программа simpleid.c . . . . .	10
4.4	Вызов программы simpleid.c . . . . .	10
4.5	Создание и компиляция simpleid2.c . . . . .	11
4.6	Программа simpleid2.c . . . . .	12
4.7	Добавление SETUID . . . . .	12
4.8	Проверка добавления SETUID-бита . . . . .	13
4.9	Выполнение программы после добавления SETUID-бита . . . . .	13
4.10	Добавление setGID-бита . . . . .	13
4.11	Выполнение программы после добавления SETUID-бита . . . . .	14
4.12	Создание и компиляция программы readfile.c . . . . .	15
4.13	Программа readfile.c . . . . .	15
4.14	Программа readfile.c . . . . .	16
4.15	Проверка запрета на чтение файла . . . . .	16
4.16	Назначение SetUID-бита . . . . .	16
4.17	Попытка прочесть файл readfile.c через программу readfile . . .	17
4.18	Попытка прочесть файл /etc/shadow через программу readfile . .	17
4.19	Проверка наличия Sticky-бита . . . . .	18
4.20	Создание и запись в файл file01.txt . . . . .	18
4.21	guest2 - чтение файла . . . . .	19
4.22	Попытка удаления файла со Sticky-битом . . . . .	20
4.23	Снятие с директории Sticky-бита . . . . .	20
4.24	Проверка снятия с директории Sticky-бита . . . . .	20
4.25	Попытка удаления файла без Sticky-бита . . . . .	21
4.26	Возвращение Sticky-бита . . . . .	21

## 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 Задание

- Изучить на практике применение SetUID- и Sticky-битов.

### 3 Теоретическое введение

**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`.

Там, где обычно установлен классический бит `x` (на исполнение), выставляется специальный бит `s`. Это позволяет обычному пользователю системы выполнять команды с повышенными привилегиями без необходимости входа в систему как `root`, разумеется зная пароль пользователя `root`. Принцип работы **Setgid** очень похож на `setuid` с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

Биты **setuid** и **setgid** выставляются с помощью команд `chmod u+s` и `chmod g+s` соответственно.[1]

**Sticky-bit** — дополнительный атрибут файлов или каталогов в операционных системах семейства UNIX. В настоящее время `sticky bit` используется в основном для каталогов, чтобы защитить в них файлы. Из такого каталога пользователь может удалить только те файлы, владельцем которых он является. Примером может служить каталог `/tmp`, в который запись открыта для всех пользователей, но нежелательно удаление чужих файлов. Установка атрибута производится утилитой `chmod`.

В операционной системе Solaris для файлов, не являющихся программами, имеет строго противоположное действие — запрещает сохранение данных этого

файла в системном кэше.[2]

Это разрешение полезно для защиты файлов от случайного удаления в среде, где несколько пользователей имеют права на запись в один и тот же каталог. Если применяется закреплённый sticky bit, пользователь может удалить файл, только если он является пользователем-владельцем файла или каталога, в котором содержится файл. По этой причине он применяется в качестве разрешения по умолчанию для каталога /tmp и может быть полезен также для каталогов общих групп.

Когда вы применяете sticky bit, пользователь может удалять файлы, только если выполняется одно из следующих условий:

- Пользователь является владельцем файла;
- Пользователь является владельцем каталога, в котором находится файл.[3]



## 4 Выполнение лабораторной работы

Прежде чем начать выполнять работу, проверим, имеются ли на устройстве все необходимые ресурсы для осуществления компиляции (рис. 4.1).

```
ovgubina@ovgubina ~]$ yum install gcc
Error: This command has to be run with superuser privileges (under the root user on most systems).
[ovgubina@ovgubina ~]$ sudo -i
[sudo] password for ovgubina:
[root@ovgubina ~]# yum install gcc
Rocky Linux 9 - BaseOS                               4.4 kB/s | 4.1 kB   00:00
Rocky Linux 9 - BaseOS                               1.4 MB/s | 1.0 MB   00:01
Rocky Linux 9 - AppStream                             9.0 kB/s | 4.5 kB   00:00
Rocky Linux 9 - AppStream                             2.6 MB/s | 7.1 MB   00:02
Rocky Linux 9 - Extras                                4.4 kB/s | 2.0 kB   00:00
Rocky Linux 9 - Extras                                16 kB/s | 11 kB    00:00
Package gcc-11.3.1-4.3.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ovgubina ~]# setenforce 0
[root@ovgubina ~]# getenforce
Permissive
[root@ovgubina ~]# whereis gcc
gcc: /usr/bin/gcc /usr/lib/gcc /usr/libexec/gcc /usr/share/man/gcc.1.gz /usr/share/info/gcc.info.gz
[root@ovgubina ~]# whereis g++
g++: /usr/bin/g++ /usr/share/man/g++.1.gz
[root@ovgubina ~]#
```

Рис. 4.1: Проверка необходимых ресурсов

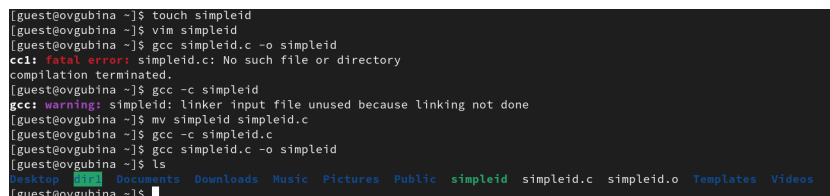
### 4.1 Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте программу simpleid.c (рис. 4.2-4.3):

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main ()
{
    uid_t uid = geteuid ();
    gid_t gid = getegid ();
```

```
printf ("uid=%d, gid=%d\n", uid, gid);
return 0;
}
```



```
[guest@ovgubina ~]$ touch simpleid
[guest@ovgubina ~]$ vim simpleid
[guest@ovgubina ~]$ gcc simpleid.c -o simpleid
cc1: fatal error: simpleid.c: No such file or directory
compilation terminated.
[guest@ovgubina ~]$ gcc -c simpleid
gcc: warning: simpleid: linker input file unused because linking not done
[guest@ovgubina ~]$ mv simpleid simpleid.c
[guest@ovgubina ~]$ gcc -c simpleid.c
[guest@ovgubina ~]$ gcc simpleid.c -o simpleid
[guest@ovgubina ~]$ ls
Desktop  Downloads  Music  Pictures  Public  simpleid  simpleid.c  simpleid.o  Templates  Videos
[guest@ovgubina ~]$
```

Рис. 4.2: Создание и компиляция файла



```
guest@ovgubina:~ -- vim simpleid
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int
main ()
{
    uid_t uid = geteuid ();
    gid_t gid = getegid ();
    printf ("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
```

Рис. 4.3: Программа simpleid.c

3. Скомпилируйте программу и убедитесь, что файл программы создан: `gcc simpleid.c -o simpleid` (рис. 4.2).

Видим, что никаких ошибок при компиляции не возникает, файлы успешно созданы (проверка через команду `ls`).

4. Выполните программу `simpleid` (рис. 4.4): `./simpleid`.



```
[guest@ovgubina ~]$ ./simpleid
uid=1001, gid=1001
[guest@ovgubina ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@ovgubina ~]$
```

Рис. 4.4: Вызов программы simpleid.c

5. Выполните системную программу `id` (рис. 4.4): `id`. и сравните полученный вами результат с данными предыдущего пункта задания.

Видим, что выводы программы, написанной ранее, и команды `id` совпадают - в каждой из них предоставляется информация об `id` владельца и группы, ими является `guest` с `uid = 1001`, `gid = 1001`.

6. Усложните программу, добавив вывод действительных идентификаторов (рис. 4.6):

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main ()
{
    uid_t real_uid = getuid ();
    uid_t e_uid = geteuid ();
    gid_t real_gid = getgid ();
    gid_t e_gid = getegid ();
    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid,
    ,→ real_gid);
    return 0;
}
```

Получившуюся программу назовите `simpleid2.c` (рис. 4.5).



```
guest@ovgubina ~$ vim simpleid2.c
guest@ovgubina ~$ gcc simpleid2.c -o simpleid2
guest@ovgubina ~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  simpleid  simpleid2  simpleid2.c  simpleid.c  simpleid.o  Templates  Videos
guest@ovgubina ~$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
guest@ovgubina ~$
```

Рис. 4.5: Создание и компиляция `simpleid2.c`

```

#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main ()
{
    uid_t real_uid = getuid ();
    uid_t e_uid = geteuid ();

    gid_t real_gid = getgid ();
    gid_t e_gid = getegid ();

    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);

    return 0;
}

```

Рис. 4.6: Программа simpleid2.c

7. Скомпилируйте и запустите simpleid2.c (рис. 4.5):

```

gcc simpleid2.c -o simpleid2
./simpleid2

```

Видим, что теперь выводятся айди владельца файла и айди текущего пользователя - сейчас это guest с uid = 1001, gid = 1001.

8. От имени суперпользователя выполните команды (рис. 4.7):

```

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

```

```

[guest@ovgubina ~]$ su ovgubina
Password:
[ovgubina@ovgubina guest]$ sudo -i
[sudo] password for ovgubina:
[root@ovgubina ~]# chown root:guest /home/guest/simpleid2
[root@ovgubina ~]# chmod u+s /home/guest/simpleid2
[root@ovgubina ~]#

```

Рис. 4.7: Добавление SETUID

9. Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды.

В данном случае я использовала `sudo` в дополнительном терминале.

Команды отличаются следующим: `su` требует пароль целевой учетной записи (например, пользователя `root`) и переключает вас на нее, в то время как `sudo` требует пароль текущего пользователя и запускает от его имени только лишь одну (или несколько) команд, на выполнение которых требуются права суперпользователя.

10. Выполните проверку правильности установки новых атрибутов и смены владельца файла `simpleid2` (рис. 4.8): `ls -l simpleid2`

```
[guest@ovgubina ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 26064 Oct  4 21:14 simpleid2
[guest@ovgubina ~]$
```

Рис. 4.8: Проверка добавления SETUID-бита

Видим добавленный SetUID-бит с правах доступа пользователя-владельца - **s**.

11. Запустите `simpleid2` и `id`:

```
./simpleid2
```

```
id
```

```
[guest@ovgubina ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@ovgubina ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@ovgubina ~]$
```

Рис. 4.9: Выполнение программы после добавления SETUID-бита

Видим, что теперь айди пользователя владельца - это айди пользователя `root`.

12. Проделайте тоже самое относительно SetGID-бита.

```
[root@ovgubina ~]# chown root:root /home/guest/simpleid2
[root@ovgubina ~]# chmod g+s /home/guest/simpleid2
[root@ovgubina ~]#
```

Рис. 4.10: Добавление setGID-бита

От пользователя guest смотрим назначение бита для группы - видим, что он добавился (рис. 4.11). Запускаем программу и видим теперь поменявшееся айди группы пользователей на айди root (рис. 4.11).

```
guest@ovgubina ~]$ ls -l simpleid2
-rwxr-sr-x. 1 root root 26064 Oct  4 21:14 simpleid2
guest@ovgubina ~]$ ./simpleid2
e_uid=1001, e_gid=0
real_uid=1001, real_gid=1001
guest@ovgubina ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
guest@ovgubina ~]$
```

Рис. 4.11: Выполнение программы после добавления SETUID-бита

### 13. Создайте программу readfile.c (рис. 4.13):

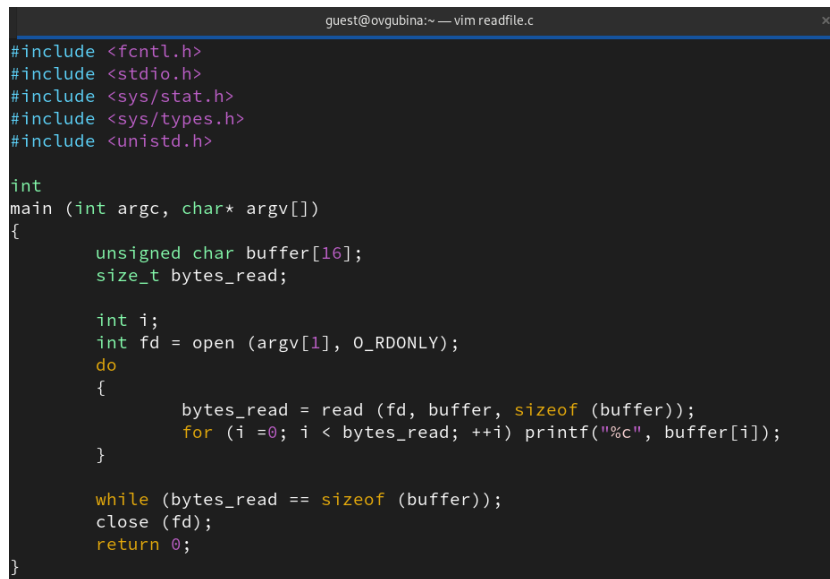
```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

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;
}
```

}

```
[guest@ovgubina ~]$ vim readfile.c
[guest@ovgubina ~]$ gcc readfile.c -o readfile
[guest@ovgubina ~]$
```

Рис. 4.12: Создание и компиляция программы readfile.c



```
guest@ovgubina:~ — vim 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. 4.13: Программа readfile.c

14. Откомпилируйте её (рис. 4.13):

```
gcc readfile.c -o readfile
```

15. Смените владельца у файла readfile.c (или любого другого текстового файла в системе) и измените права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

Владельца файла меняем на root, назначаем права доступа 733 - дают полные права пользователю-владельцу и права на запись и выполнение остальным пользователям и группе (рис. 4.14).

```
[root@ovgubina ~]# chown root:guest /home/guest/readfile.c
[root@ovgubina ~]# chmod 733 /home/guest/readfile.c
[root@ovgubina ~]#
```

Рис. 4.14: Программа readfile.c

16. Проверьте, что пользователь guest не может прочитать файл readfile.c (рис. 4.15).

```
[guest@ovgubina ~]$ ls -l readfile.c
-rwx-wx-wx. 1 root guest 421 Oct  4 21:30 readfile.c
[guest@ovgubina ~]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@ovgubina ~]$
```

Рис. 4.15: Проверка запрета на чтение файла

Пользователь действительно не может прочитать файл readfile.c - выводится сообщение **Permission denied**.

17. Смените у программы readfile владельца и установите SetUID-бит.

Меняем владельца на root и назначаем SetUID-бит (рис. 4.16).

```
[root@ovgubina ~]# chown root:guest /home/guest/readfile
[root@ovgubina ~]# chmod u+s /home/guest/readfile
[root@ovgubina ~]#
```

Рис. 4.16: Назначение SetUID-бита

18. Проверьте, может ли программа readfile прочитать файл readfile.c (рис. 4.16)?



```
[guest@ovgubina ~]$ ./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;
}

[guest@ovgubina ~]$
```

Рис. 4.17: Попытка прочитать файл readfile.c через программу readfile

Нам удалось прочитать файл.

19. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отрадите полученный результат и ваши объяснения в отчёте.

```
[guest@ovgubina ~]$ ./readfile /etc/shadow
root:$6$sEldG2BtcsE5HK.$McENz/OtucocM2T7.jTwNAa488m9HOZJW.I0wfInQs6X/LTcQv7ph0LLgKtNGc9h7nESgphzL5XRH6R157uE0:0:99999:7:::
bin:*:19469:0:99999:7:::
daemon:*:19469:0:99999:7:::
adm:*:19469:0:99999:7:::
lp:*:19469:0:99999:7:::
sync:*:19469:0:99999:7:::
shutdown:*:19469:0:99999:7:::
halt:*:19469:0:99999:7:::
mail:*:19469:0:99999:7:::
operator:*:19469:0:99999:7:::
games:*:19469:0:99999:7:::
ftp:*:19469:0:99999:7:::
nobody:*:19469:0:99999:7:::
systemd-coredump:!:19607:!!!!:
dbus:!:19607:!!!!:
polkitd:!:19607:!!!!:
avahi:!:19607:!!!!:
rtkit:!:19607:!!!!:
sssd:!:19607:!!!!:
pipewire:!:19607:!!!!:
libstoragemgmt:!:19607:!!!!:
systemd-oom:!:19607:!!!!:
tss:!:19607:!!!!:
geoclue:!:19607:!!!!:
cockpit-ws:!:19607:!!!!:
cockpit-wsinstance:!:19607:!!!!:
flatpak:!:19607:!!!!:
colord:!:19607:!!!!:
clevi:!:19607:!!!!:
setroubleshoot:!:19607:!!!!:
gdm:!:19607:!!!!:
pesign:!:19607:!!!!:
```

Рис. 4.18: Попытка прочитать файл /etc/shadow через программу readfile

Нам удалось прочитать оба файла поскольку мы делали это с помощью программы readfile, которая принадлежит суперпользователю root.

## 4.2 Исследование Sticky-бита

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду `ls -l / | grep tmp` (рис. 4.19).

```
[root@ovgubina ~]# ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 Oct  4 21:47 tmp
[root@ovgubina ~]#
```

Рис. 4.19: Проверка наличия Sticky-бита

2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt` (рис. 4.20)

```
[guest@ovgubina ~]$ echo "test" > /tmp/file01.txt
[guest@ovgubina ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Oct  4 21:49 /tmp/file01.txt
[guest@ovgubina ~]$ chmod o+rw /tmp/file01.txt
[guest@ovgubina ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Oct  4 21:49 /tmp/file01.txt
[guest@ovgubina ~]$
```

Рис. 4.20: Создание и запись в файл file01.txt

3. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные» (рис. 4.20):

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

4. От пользователя guest2 (не являющегося владельцем) попробуйте прочесть файл /tmp/file01.txt: `cat /tmp/file01.txt` (рис. 4.21).

```
[guest@ovgubina ~]$ su guest2
Password:
[guest2@ovgubina guest]$ cat /tmp/file01.txt
test
[guest2@ovgubina guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@ovgubina guest]$ cat /tmp/file01.txt
test
[guest2@ovgubina guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@ovgubina guest]$ cat /tmp/file01.txt
test
[guest2@ovgubina guest]$
```

Рис. 4.21: guest2 - чтение файла

Содержимое файла выводится - можем прочитать файл.

5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" >> /tmp/file01.txt` (рис. 4.21). Удалось ли вам выполнить операцию?

Выполнить операцию не удалось - `Permission denied`.

6. Проверьте содержимое файла командой `cat /tmp/file01.txt` (рис. 4.21).

Файл никак не изменился.

7. От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой `echo "test3" > /tmp/file01.txt` (рис. 4.21). Удалось ли вам выполнить операцию?

Выполнить операцию не удалось - `Permission denied`.

8. Проверьте содержимое файла командой `cat /tmp/file01.txt` (рис. 4.21).

Файл никак не изменился.

9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt` (рис. 4.22). Удалось ли вам удалить файл?

```
[guest2@ovgubina guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@ovgubina guest]$
```

Рис. 4.22: Попытка удаления файла со Sticky-битом

10. Повысьте свои права до суперпользователя следующей командой `su -` и выполните после этого команду, снимающую атрибут `t` (Sticky-бит) с директории `/tmp`: `chmod -t /tmp` (рис. 4.23).

```
[guest2@ovgubina guest]$ su -
Password:
[root@ovgubina ~]# chmod -t /tmp
[root@ovgubina ~]# exit
logout
[guest2@ovgubina guest]$
```

Рис. 4.23: Снятие с директории Sticky-бита

11. Покиньте режим суперпользователя командой `exit` (рис. 4.23).
12. От пользователя `guest2` проверьте, что атрибута `t` у директории `/tmp` нет:  
`ls -l / | grep tmp` (рис. 4.24)

```
[guest2@ovgubina guest]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 Oct  4 21:54 tmp
[guest2@ovgubina guest]$ cat /tmp/file01.txt
test
[guest2@ovgubina guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@ovgubina guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@ovgubina guest]$ cat /tmp/file01.txt
test
```

Рис. 4.24: Проверка снятия с директории Sticky-бита

13. Повторите предыдущие шаги (рис. 4.24). Какие наблюдаются изменения?

Изменений не наблюдается, мы также можем читать файл, но не можем производить дозапись и запись в него.

14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем (рис. 4.25)? Ваши наблюдения занесите в отчёт.

```
[guest2@ovgubina guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
[guest2@ovgubina guest]$
```

Рис. 4.25: Попытка удаления файла без Sticky-бита

В этот раз без Sticky-бита нам удалось успешно удалить файл file01.txt.

15. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp (рис. 4.26):

```
su -
chmod +t /tmp
exit
```

```
[guest2@ovgubina guest]$ su -
Password:
[root@ovgubina ~]# chmod +t /tmp
[root@ovgubina ~]# exit
logout
[guest2@ovgubina guest]$
```

Рис. 4.26: Возвращение Sticky-бита

## 5 Выводы

Изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получила практические навыки работы в консоли с дополнительными атрибутами. Рассмотрела работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

## Список литературы

1. Использование SETUID, SETGID и Sticky bit для расширенной настройки прав доступа в операционных системах Linux [Электронный ресурс]. 2023. URL: <https://ruvds.com/ru/helpcenter/suid-sgid-sticky-bit-linux/>.
2. Sticky bit [Электронный ресурс]. 2023. URL: [https://ru.wikipedia.org/wiki/Sticky\\_bit](https://ru.wikipedia.org/wiki/Sticky_bit).
3. Права в Linux (chown, chmod, SUID, GUID, sticky bit, ACL, umask) [Электронный ресурс]. 2023. URL: <https://habr.com/ru/articles/469667/>.