

# **Лабораторная работа №5**

**Дисциплина: основы информационной безопасности**

Астраханцева А. А.

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
3.1	Подготовка лабораторного стенда . . . . .	8
3.2	Создание программы . . . . .	8
3.3	Исследование Sticky-бита . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>18</b>
<b>5</b>	<b>Список литературы. Библиография</b>	<b>19</b>

## Список иллюстраций

3.1	Подготовка лабораторного стенда . . . . .	8
3.2	Создание программы simpleid.c . . . . .	9
3.3	Компиляция и запуск программы simpleid.c . . . . .	10
3.4	Создание программы simpleid2.c . . . . .	11
3.5	Компиляция программы simpleid2.c . . . . .	11
3.6	Изменение владельца и прав доступа для simpleid2 . . . . .	12
3.7	Создание программы readfile.c . . . . .	13
3.8	Пункты 14-19 . . . . .	14
3.9	Проверка установки Sticky бита на директории /tmp, создание нового файла, изменение прав доступа к нему . . . . .	15
3.10	От лица guest2 попытка прочесть файл, изменить его и удалить его	16
3.11	Переход в режим суперпользователя, снятие Sticky бита . . . . .	16
3.12	Повторение пунктов 4-9 со снятым Sticky битом . . . . .	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

### SetUID

Бит доступа SETUID дает возможность запускать файл на исполнение от имени владельца файла. Например, именно так работает программа `sudo`. Если мы посмотрим на нее через `ls`, то увидим что вместо ключа `x` стоит ключ `s`. Это и означает, что у данного файла включен или выставлен параметр SETUID. А значит, если обычный пользователь запустит этот файл, то файл будет запущен от имени владельца, то есть `root`. Важно понимать, что никакого ввода пароля при этом не потребуется. А сам запрос и обработка пароля организованы уже внутри программы `sudo` [1].

### Sticky bit

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 закрепленный 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.

Без sticky bit, если пользователь может создавать файлы в каталоге, он также может удалять файлы из этого каталога. В общедоступной групповой среде это может раздражать. Представьте себе пользователей `linda` и `lori`, которые оба имеют права на запись в каталог `/data/account` и получают эти разрешения благодаря участию в группе `account`. Поэтому `linda` может удалять файлы, созданные `lori`, и

наоборот.

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.

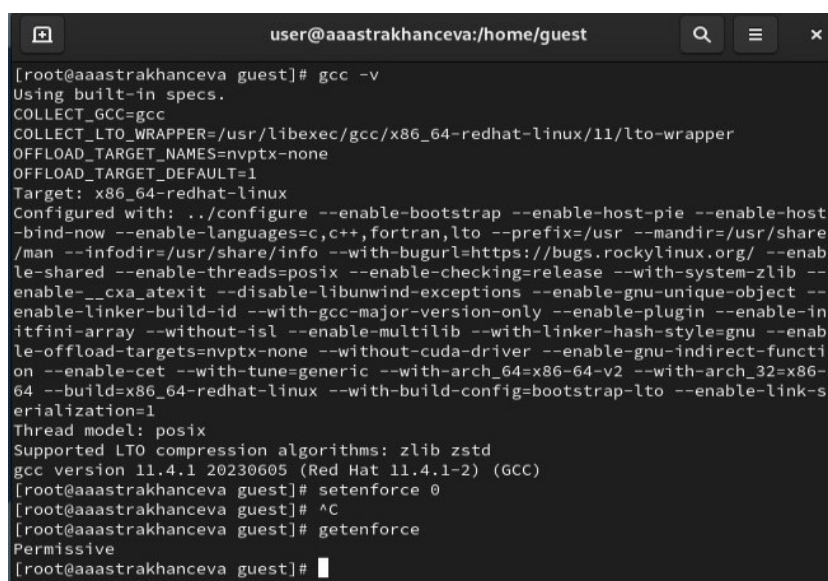
При использовании `ls -ld`, вы можете видеть sticky bit как `t` в позиции, где вы обычно видите разрешение на выполнение для других [2].

## 3 Выполнение лабораторной работы

### 3.1 Подготовка лабораторного стенда

Проверяем, установлен ли у нас компилятор gcc командой `gcc -v`.

Отключаем систему запретов до очередной перезагрузки системы командой `setenforce 0`. После этого команда `getenforce` должна выводить `Permissive` (рис. 3.1)

A terminal window titled 'user@aaastrakhanceva:/home/guest' showing the output of the command 'gcc -v'. The output displays the GCC version 11.4.1 and its configuration options. Below this, the command 'setenforce 0' is executed, followed by 'getenforce', which returns 'Permissive'.

```
[root@aaastrakhanceva guest]# gcc -v
Using built-in specs.
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
Target: x86_64-redhat-linux
Configured with: ../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=https://bugs.rockylinux.org/ --enab
le-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-in
itfini-array --without-isl --enable-multilib --with-linker-hash-style=gnu --enab
le-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-functi
on --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-s
erialization=1
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)
[root@aaastrakhanceva guest]# setenforce 0
[root@aaastrakhanceva guest]# ^C
[root@aaastrakhanceva guest]# getenforce
Permissive
[root@aaastrakhanceva guest]#
```

Рис. 3.1: Подготовка лабораторного стенда

### 3.2 Создание программы

1. Войдите в систему от имени пользователя guest.



## 2. Создайте программу simpleid.c: (рис. 3.2)

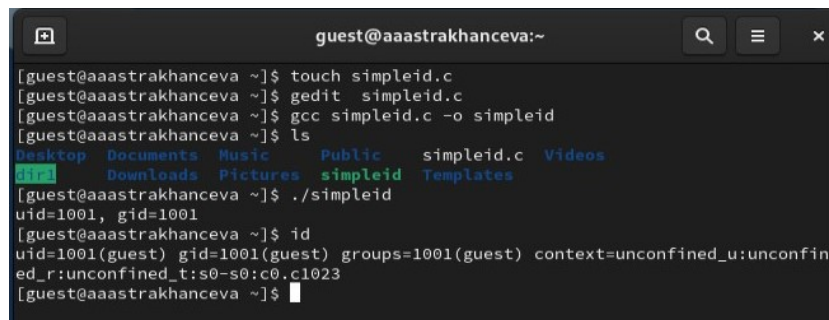
```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main ()
{
    uid_t uid = geteuid ();
    gid_t gid = getegid ();
    printf ("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
```



Рис. 3.2: Создание программы simpleid.c

3. Скомпилируйте программу и убедитесь, что файл программы создан: `gcc simpleid.c -o simpleid`
4. Выполните программу simpleid: `./simpleid`
5. Выполните системную программу id: `id` и сравните полученный вами результат с данными предыдущего пункта задания. Выводы команд совпали (рис. 3.3).

A terminal window titled 'guest@aaastrakhanceva:~' with search, menu, and close buttons. The terminal shows the following commands and output:

```
[guest@aaastrakhanceva ~]$ touch simpleid.c
[guest@aaastrakhanceva ~]$ gedit simpleid.c
[guest@aaastrakhanceva ~]$ gcc simpleid.c -o simpleid
[guest@aaastrakhanceva ~]$ ls
Desktop  Documents  Music      Public     simpleid.c  Videos
dir      Downloads  Pictures   simpleid   Templates
[guest@aaastrakhanceva ~]$ ./simpleid
uid=1001, gid=1001
[guest@aaastrakhanceva ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aaastrakhanceva ~]$
```

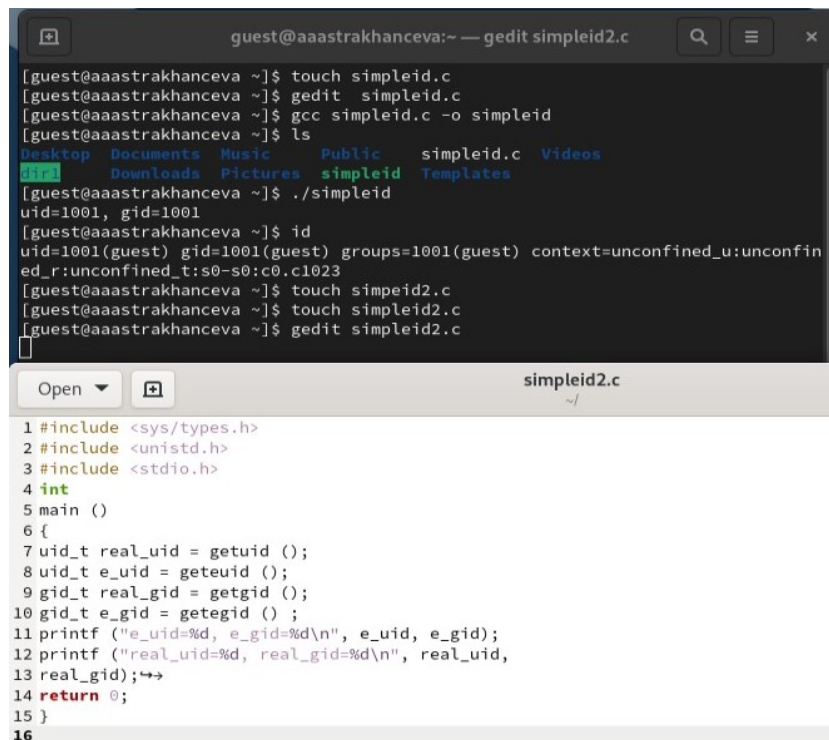
Рис. 3.3: Компиляция и запуск программы simpleid.c

6. Усложните программу, добавив вывод действительных идентификаторов:

```
#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. 3.4).

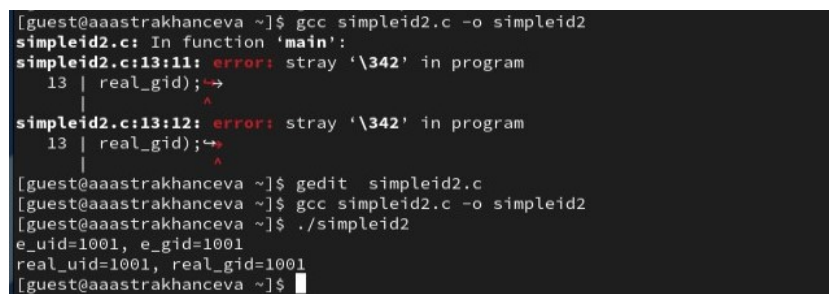


```
guest@aaastrakhanceva: ~ — gedit simpleid2.c
[guest@aaastrakhanceva ~]$ touch simpleid.c
[guest@aaastrakhanceva ~]$ gedit simpleid.c
[guest@aaastrakhanceva ~]$ gcc simpleid.c -o simpleid
[guest@aaastrakhanceva ~]$ ls
Desktop Documents Music Public simpleid.c Videos
simpleid Downloads Pictures simpleid Templates
[guest@aaastrakhanceva ~]$ ./simpleid
uid=1001, gid=1001
[guest@aaastrakhanceva ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aaastrakhanceva ~]$ touch simpleid2.c
[guest@aaastrakhanceva ~]$ touch simpleid2.c
[guest@aaastrakhanceva ~]$ gedit simpleid2.c

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,
13    real_gid);↵
14    return 0;
15 }
16
```

Рис. 3.4: Создание программы simpleid2.c

7. Скомпилируйте и запустите simpleid2.c: `gcc simpleid2.c -o simpleid2, ./simpleid2` (рис. 3.5).

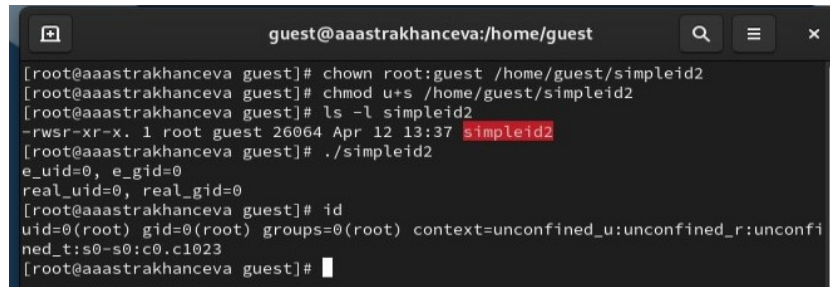


```
[guest@aaastrakhanceva ~]$ gcc simpleid2.c -o simpleid2
simpleid2.c: In function 'main':
simpleid2.c:13:11: error: stray '\342' in program
   13 | real_gid);↵
      |           ^
simpleid2.c:13:12: error: stray '\342' in program
   13 | real_gid);↵
      |           ^
[guest@aaastrakhanceva ~]$ gedit simpleid2.c
[guest@aaastrakhanceva ~]$ gcc simpleid2.c -o simpleid2
[guest@aaastrakhanceva ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@aaastrakhanceva ~]$
```

Рис. 3.5: Компиляция программы simpleid2.c

8. От имени суперпользователя выполните команды: `chown root:guest /home/guest/simpleid2, chmod u+s /home/guest/simpleid2`
9. Используйте `sudo` или повысьте временно свои права с помощью `su`. Поясните, что делают эти команды. Первая команда меняет владельца файла simpleid2.c, а вторая устанавливает на этот файл SetUID бит.

10. Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2: `ls -l simpleid2`
11. Запустите simpleid2 и id: `./simpleid2`, id Сравните результаты. (рис. 3.6).



```
guest@aaastrakhanceva:/home/guest
[root@aaastrakhanceva guest]# chown root:guest /home/guest/simpleid2
[root@aaastrakhanceva guest]# chmod u+s /home/guest/simpleid2
[root@aaastrakhanceva guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 26064 Apr 12 13:37 simpleid2
[root@aaastrakhanceva guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@aaastrakhanceva guest]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfi
ned_t:s0-s0:c0.c1023
[root@aaastrakhanceva guest]#
```

Рис. 3.6: Изменение владельца и прав доступа для simpleid2

12. Создайте программу readfile.c: (рис. 3.7).

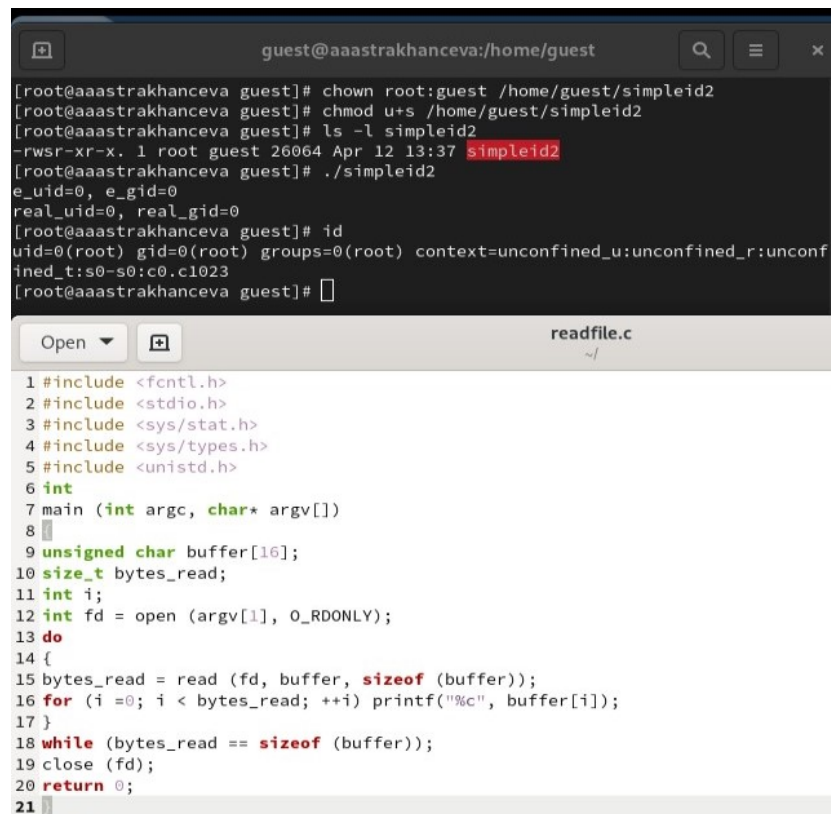
```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

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;
}

```



The screenshot shows a terminal window at the top and a code editor at the bottom. The terminal window title is 'guest@aaastrakhanceva:/home/guest'. It shows the following commands and output:

```

[root@aaastrakhanceva guest]# chown root:guest /home/guest/simpleid2
[root@aaastrakhanceva guest]# chmod u+s /home/guest/simpleid2
[root@aaastrakhanceva guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 26064 Apr 12 13:37 simpleid2
[root@aaastrakhanceva guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@aaastrakhanceva guest]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@aaastrakhanceva guest]#

```

The code editor window title is 'readfile.c'. It contains the following C code:

```

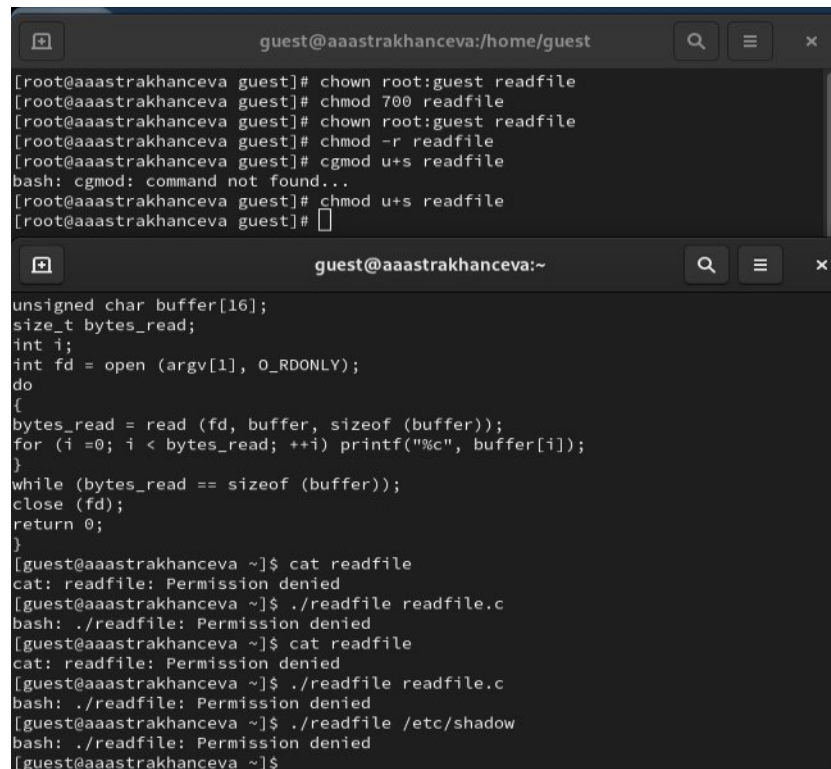
1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <sys/stat.h>
4 #include <sys/types.h>
5 #include <unistd.h>
6 int
7 main (int argc, char* argv[])
8 {
9     unsigned char buffer[16];
10    size_t bytes_read;
11    int i;
12    int fd = open (argv[1], O_RDONLY);
13    do
14    {
15        bytes_read = read (fd, buffer, sizeof (buffer));
16        for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
17    }
18    while (bytes_read == sizeof (buffer));
19    close (fd);
20    return 0;
21

```

Рис. 3.7: Создание программы readfile.c

14. Откомпилируйте её. `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.
16. Проверьте, что пользователь guest не может прочитать файл readfile.c.
17. Смените у программы readfile владельца и установите SetU'D-бит.

18. Проверьте, может ли программа readfile прочитать файл readfile.c?
19. Проверьте, может ли программа readfile прочитать файл /etc/shadow? (рис. 3.8).



```
guest@aaastrakhanceva:/home/guest
[root@aaastrakhanceva guest]# chown root:guest readfile
[root@aaastrakhanceva guest]# chmod 700 readfile
[root@aaastrakhanceva guest]# chown root:guest readfile
[root@aaastrakhanceva guest]# chmod -r readfile
[root@aaastrakhanceva guest]# chmod u+s readfile
bash: chmod: command not found...
[root@aaastrakhanceva guest]# chmod u+s readfile
[root@aaastrakhanceva guest]#

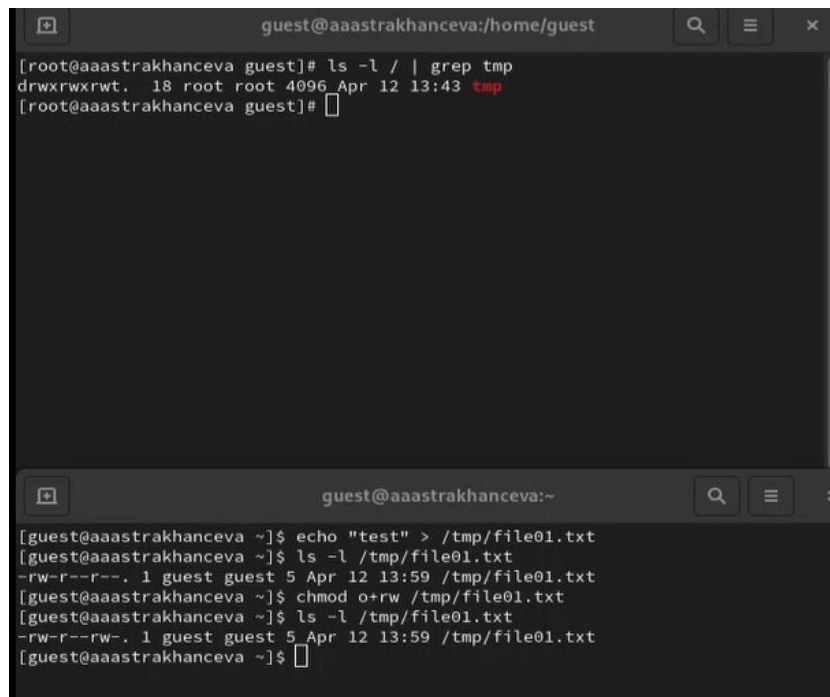
guest@aaastrakhanceva:~
unsigned char buffer[16];
size_t bytes_read;
int i;
int fd = open (argv[1], O_RDONLY);
do
{
bytes_read = read (fd, buffer, sizeof (buffer));
for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
}
while (bytes_read == sizeof (buffer));
close (fd);
return 0;
}
[guest@aaastrakhanceva ~]$ cat readfile
cat: readfile: Permission denied
[guest@aaastrakhanceva ~]$ ./readfile readfile.c
bash: ./readfile: Permission denied
[guest@aaastrakhanceva ~]$ cat readfile
cat: readfile: Permission denied
[guest@aaastrakhanceva ~]$ ./readfile readfile.c
bash: ./readfile: Permission denied
[guest@aaastrakhanceva ~]$ ./readfile /etc/shadow
bash: ./readfile: Permission denied
[guest@aaastrakhanceva ~]$
```

Рис. 3.8: Пункты 14-19

### 3.3 Исследование Sticky-бита

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду `ls -l / | grep tmp`
2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt`
3. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные»: `ls -l`

/tmp/file01.txt, chmod o+rw /tmp/file01.txt, ls -l /tmp/file01.txt (рис. 3.9).



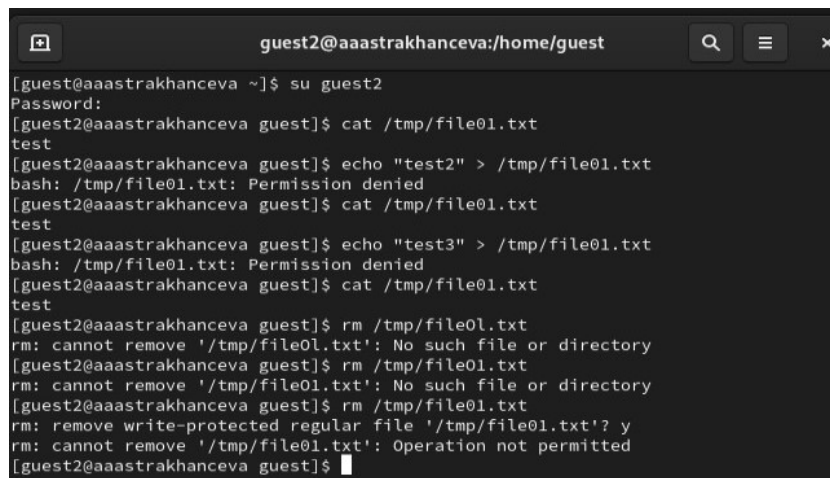
```
guest@aaastrakhanceva:/home/guest
[root@aaastrakhanceva guest]# ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 Apr 12 13:43 tmp
[root@aaastrakhanceva guest]#

guest@aaastrakhanceva:~
[guest@aaastrakhanceva ~]$ echo "test" > /tmp/file01.txt
[guest@aaastrakhanceva ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Apr 12 13:59 /tmp/file01.txt
[guest@aaastrakhanceva ~]$ chmod o+rw /tmp/file01.txt
[guest@aaastrakhanceva ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Apr 12 13:59 /tmp/file01.txt
[guest@aaastrakhanceva ~]$
```

Рис. 3.9: Проверка установки Sticky бита на дирркетории /tmp, создание нового файла, изменение пра доступа к нему

4. От пользователя guest2 (не являющегося владельцем) попробуйте прочит-  
тать файл /tmp/file01.txt: `cat /tmp/file01.txt`
5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово  
test2 командой `echo "test2" > /tmp/file01.txt`. Удалось ли вам выпол-  
нить операцию?
6. Проверьте содержимое файла командой `cat /tmp/file01.txt`
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`. Удалось ли вам выполнить операцию?
8. Проверьте содержимое файла командой `cat /tmp/file01.txt`

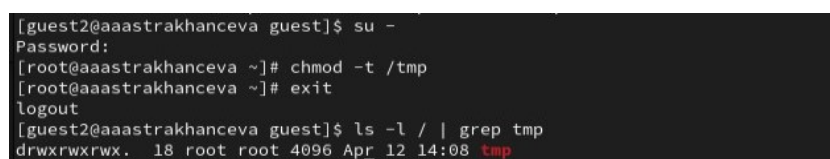
9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`. Удалось ли вам удалить файл? (рис. 3.10).



```
guest2@aaastrakhanceva:/home/guest
[guest@aaastrakhanceva ~]$ su guest2
Password:
[guest2@aaastrakhanceva guest]$ cat /tmp/file01.txt
test
[guest2@aaastrakhanceva guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aaastrakhanceva guest]$ cat /tmp/file01.txt
test
[guest2@aaastrakhanceva guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aaastrakhanceva guest]$ cat /tmp/file01.txt
test
[guest2@aaastrakhanceva guest]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory
[guest2@aaastrakhanceva guest]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory
[guest2@aaastrakhanceva 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@aaastrakhanceva guest]$
```

Рис. 3.10: От лица guest2 попытка прочесть файл, изменить его и удалить его

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`
11. Покиньте режим суперпользователя командой `exit`
12. От пользователя guest2 проверьте, что атрибута `t` у директории `/tmp` нет: `ls -l / | grep tmp` (рис. 3.11).



```
[guest2@aaastrakhanceva guest]$ su -
Password:
[root@aaastrakhanceva ~]# chmod -t /tmp
[root@aaastrakhanceva ~]# exit
logout
[guest2@aaastrakhanceva guest]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 Apr 12 14:08 tmp
```

Рис. 3.11: Переход в режим суперпользователя, снятие Sticky бита

13. Повторите предыдущие шаги. Какие наблюдаются изменения?
14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Ваши наблюдения занесите в отчёт. Внести изменения в файл не удалось, зато получилось его удалить.



15. Повысьте свои права до суперпользователя и верните атрибут `t` на директорию `/tmp`: `su - chmod +t /tmp exit` (рис. 3.12).

```
[guest2@aaastrakhanceva guest]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 Apr 12 14:08 tmp
[guest2@aaastrakhanceva guest]$ cat /tmp/file01.txt
test
[guest2@aaastrakhanceva guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aaastrakhanceva guest]$ cat /tmp/file01.txt
test
[guest2@aaastrakhanceva guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aaastrakhanceva guest]$ cat /tmp/file01.txt
test
[guest2@aaastrakhanceva guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
[guest2@aaastrakhanceva guest]$ ls /tmp
systemd-private-bd9ab20792b04d54ae93f80037242902-chrond.service-fDRJsn
systemd-private-bd9ab20792b04d54ae93f80037242902-colord.service-CkEISg
systemd-private-bd9ab20792b04d54ae93f80037242902-dbus-broker.service-WzEajM
systemd-private-bd9ab20792b04d54ae93f80037242902-fwupd.service-DIqn3I
systemd-private-bd9ab20792b04d54ae93f80037242902-kdump.service-AmdVnn
systemd-private-bd9ab20792b04d54ae93f80037242902-ModemManager.service-8Pjibz
systemd-private-bd9ab20792b04d54ae93f80037242902-power-profiles-daemon.service-Bt5C3Z
systemd-private-bd9ab20792b04d54ae93f80037242902-rtkit-daemon.service-yWzqqS
systemd-private-bd9ab20792b04d54ae93f80037242902-switcheroo-control.service-GrhEGF
systemd-private-bd9ab20792b04d54ae93f80037242902-systemd-logind.service-KimgXH
systemd-private-bd9ab20792b04d54ae93f80037242902-upower.service-YXTirC
tmp-f57bb7af-56b2-4ef1-8ce4-50bfb4fc10e1
[guest2@aaastrakhanceva guest]$ cd /tmp | grep file01.txt
```

Рис. 3.12: Повторение пунктов 4-9 со снятым Sticky битом

## 4 Выводы

В ходе выполнения лабораторной работы я изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Также 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.

## 5 Список литературы. Библиография

[1] Права доступа на файлы. Часть 3. Специальные биты разрешений SETUID, SETGID и Sticky bit: <https://dzen.ru/a/Y-S4apsOlAJ73hJe>

[2] Права в Linux (chown, chmod, SUID, GUID, sticky bit, ACL, umask): <https://habr.com/ru/articles/469667/>