

Отчет по лабораторной работе №5

**Дискреционное разграничение прав Linux. Исследования влияния
расширенных атрибутов**

Гисматуллин Артём Вадимович НПИбд-01-22

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Создание программы	7
3.2	Исследование Stiky-бита	13
4	Выводы	16
	Список литературы	17

Список иллюстраций

3.1	Редактор. Файл simpleid.c	7
3.2	Командная строка. Сравнение с выводом id	8
3.3	Редактор. Новая программа	9
3.4	Командная строка. Изменение владельца и атрибутов файла . . .	10
3.5	Командная строка. Сравнение результатов вывода	11
3.6	Редактор. Файл readfile	12
3.7	Командная строка. Проверка изменений	13
3.8	Командная строка. Исследование Sticky-бита	14
3.9	Командная строка. Исследование Sticky-бита	15

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

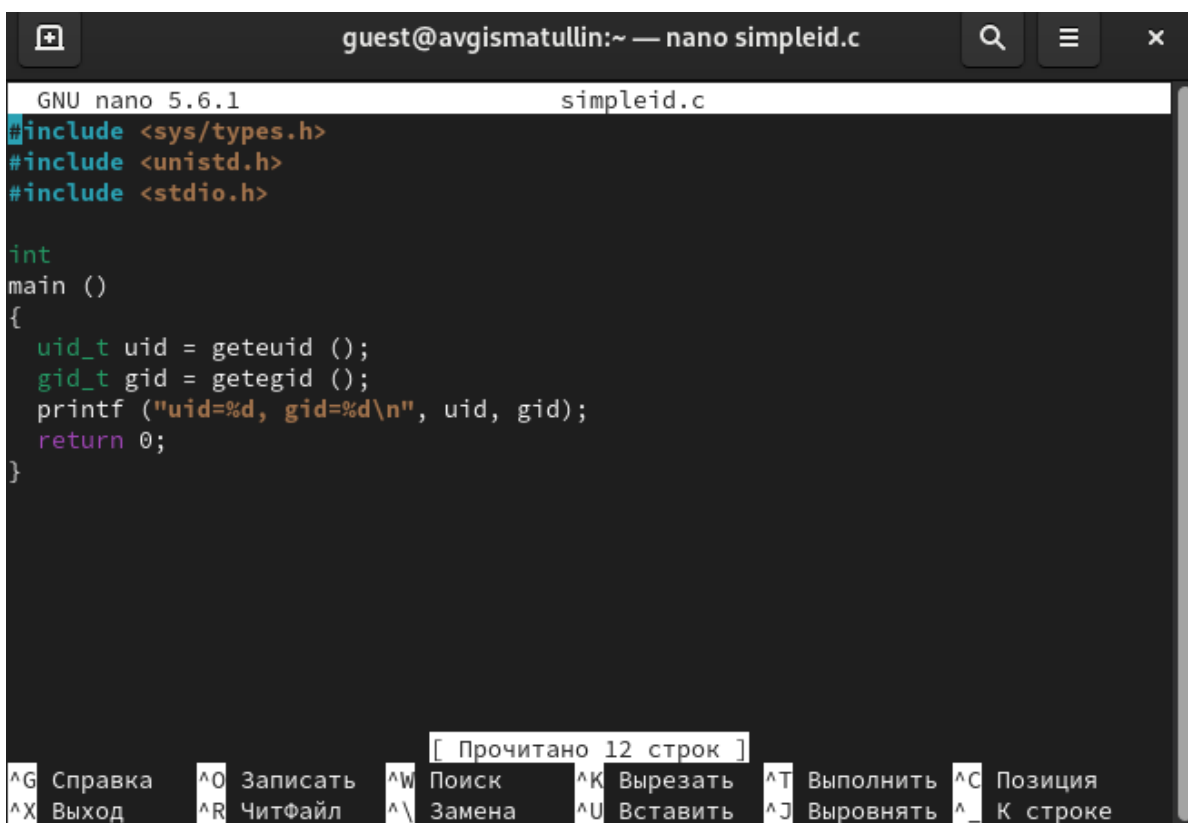
2 Задание

Последовательно выполнять все пункты, занося ответы и замечания в отчет.

3 Выполнение лабораторной работы

3.1 Создание программы

1. Зайдем под учетной записью guest и создадим первую программу под названием simpleid.c (рис. 3.1)



```
GNU nano 5.6.1 simpleid.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

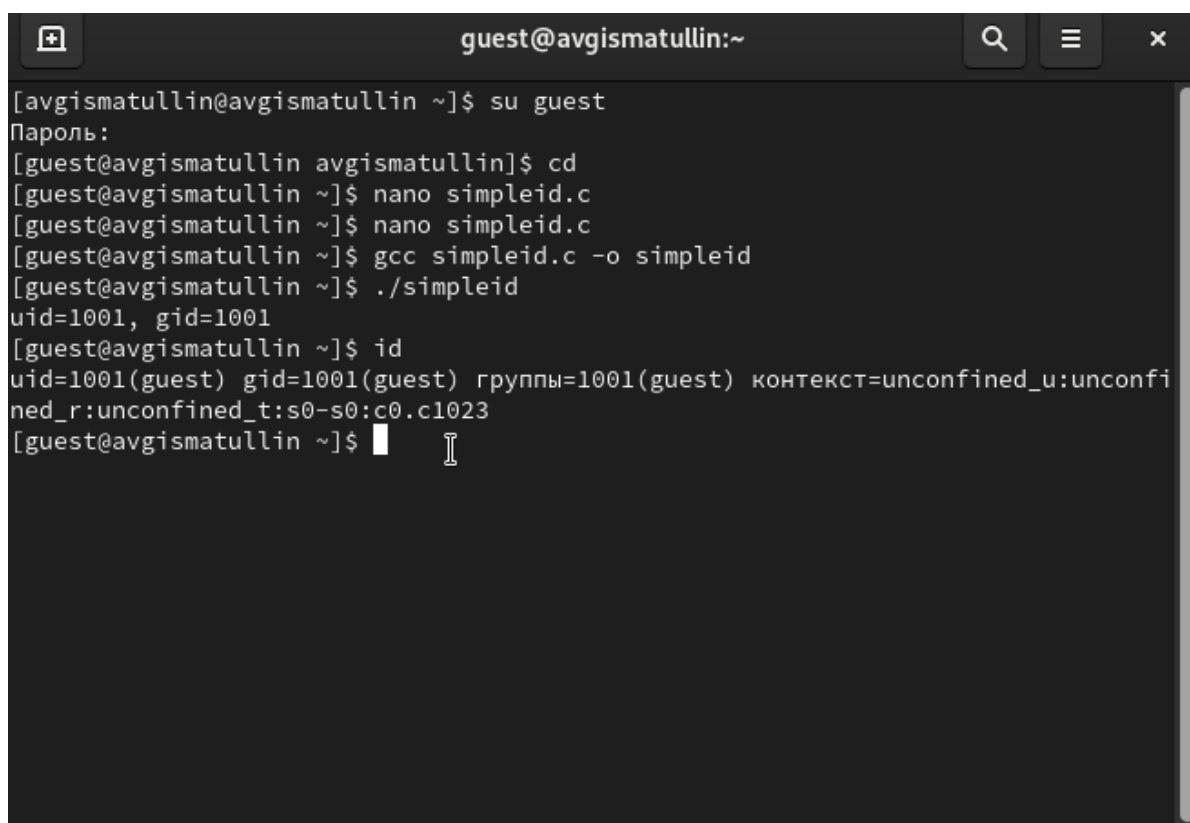
int
main ()
{
    uid_t uid = getuid ();
    gid_t gid = getegid ();
    printf ("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
```

[Прочитано 12 строк]

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^I Замена	^U Вставить	^J Выводить	^_ К строке

Рис. 3.1: Редактор. Файл simpleid.c

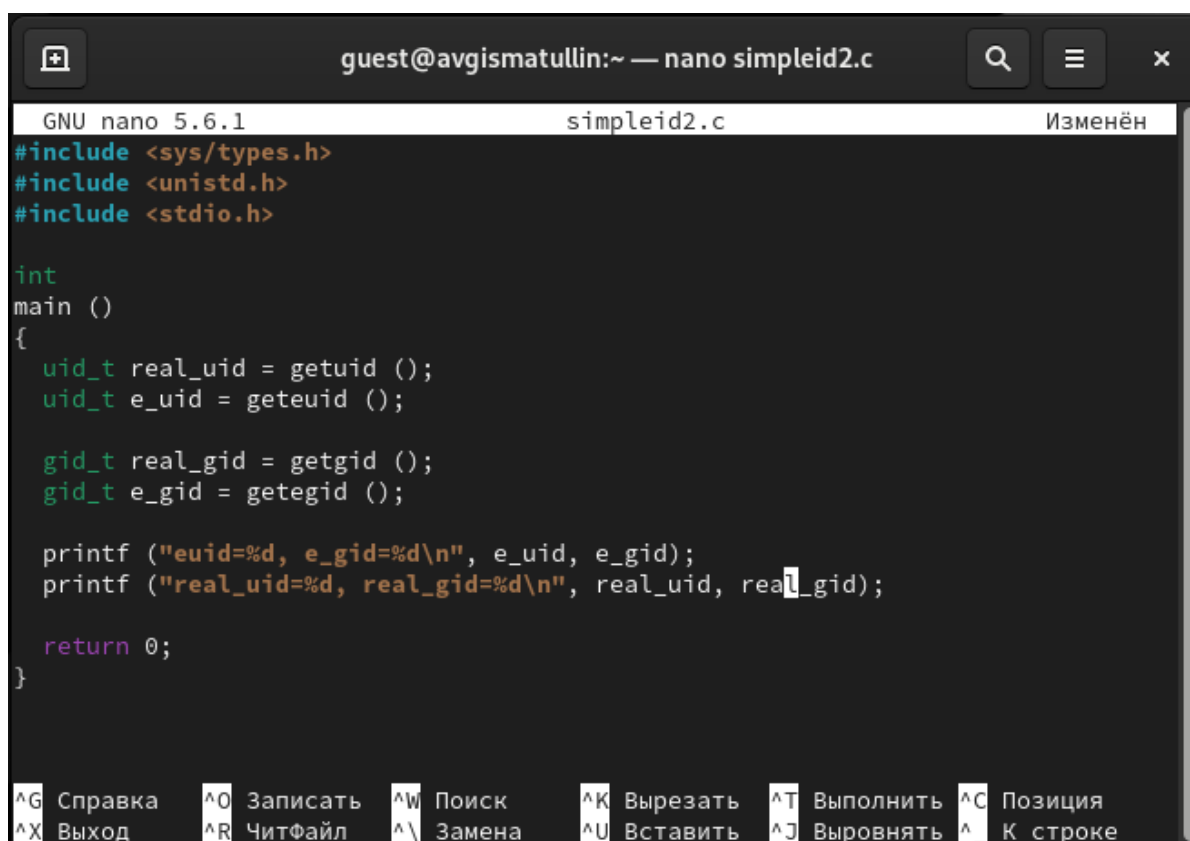
2. Скомпилируем программу командой `gcc simpleid.c -o simpleid`, а затем выполним программу. Сравним с системным выводом `id` (рис. 3.2)

A terminal window titled 'guest@avgismatullin:~' with search, menu, and close icons in the title bar. The terminal shows a sequence of commands: 'su guest' (password prompt), 'cd', 'nano simpleid.c' (twice), 'gcc simpleid.c -o simpleid', and './simpleid'. The output of './simpleid' is 'uid=1001, gid=1001'. Then, the 'id' command is run, showing 'uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023'.

```
guest@avgismatullin:~  
[avgismatullin@avgismatullin ~]$ su guest  
Пароль:  
[guest@avgismatullin avgismatullin]$ cd  
[guest@avgismatullin ~]$ nano simpleid.c  
[guest@avgismatullin ~]$ nano simpleid.c  
[guest@avgismatullin ~]$ gcc simpleid.c -o simpleid  
[guest@avgismatullin ~]$ ./simpleid  
uid=1001, gid=1001  
[guest@avgismatullin ~]$ id  
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi  
ned_r:unconfined_t:s0-s0:c0.c1023  
[guest@avgismatullin ~]$
```

Рис. 3.2: Командная строка. Сравнение с выводом id

3. Далее создадим файл simpleid2.c и в дальнейшем скомпилируем его (рис. 3.3)



```
GNU nano 5.6.1 simpleid2.c Изменён
#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 ();

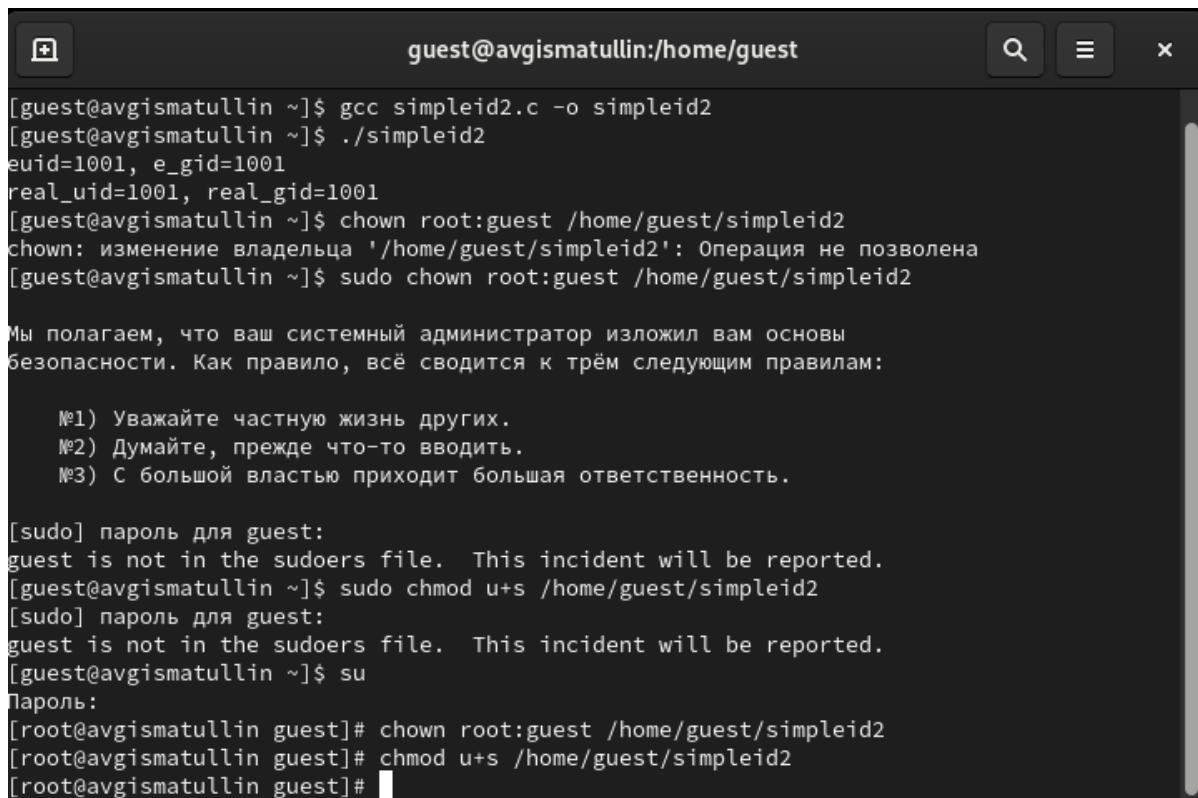
    printf ("euid=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);

    return 0;
}
```

^G Справка **^O** Записать **^W** Поиск **^K** Вырезать **^T** Выполнить **^C** Позиция
^X Выход **^R** ЧитФайл **^_** Замена **^U** Вставить **^J** Выровнять **^_** К строке

Рис. 3.3: Редактор. Новая программа

- После этого (после компиляции) запустим его и пропишем команды по смене владельца файла simpleid2. При помощи команды `ls -l simpleid2` проверим установку новых атрибутов и владельца, а затем сравним вывод программы simpleid2 и `id`. Заметим, что выводы отличаются (рис. 3.4)



```
guest@avgismatullin:/home/guest

[guest@avgismatullin ~]$ gcc simpleid2.c -o simpleid2
[guest@avgismatullin ~]$ ./simpleid2
euid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@avgismatullin ~]$ chown root:guest /home/guest/simpleid2
chown: изменение владельца '/home/guest/simpleid2': Операция не позволена
[guest@avgismatullin ~]$ sudo chown root:guest /home/guest/simpleid2

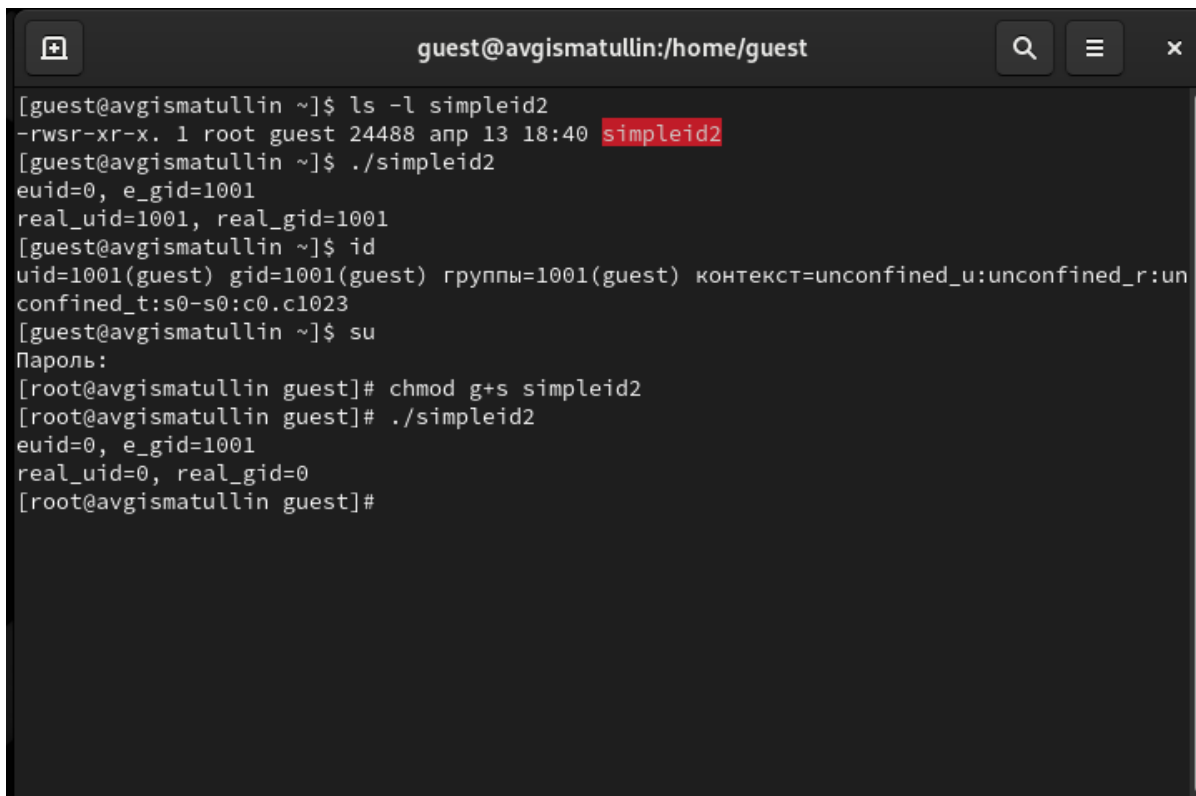
Мы полагаем, что ваш системный администратор изложил вам основы
безопасности. Как правило, всё сводится к трём следующим правилам:

    №1) Уважайте частную жизнь других.
    №2) Думайте, прежде что-то вводить.
    №3) С большой властью приходит большая ответственность.

[sudo] пароль для guest:
guest is not in the sudoers file. This incident will be reported.
[guest@avgismatullin ~]$ sudo chmod u+s /home/guest/simpleid2
[sudo] пароль для guest:
guest is not in the sudoers file. This incident will be reported.
[guest@avgismatullin ~]$ su
Пароль:
[root@avgismatullin guest]# chown root:guest /home/guest/simpleid2
[root@avgismatullin guest]# chmod u+s /home/guest/simpleid2
[root@avgismatullin guest]#
```

Рис. 3.4: Командная строка. Изменение владельца и атрибутов файла

В самом конце проделаем то же самое относительно SetGID-бита (рис. 3.5)



```
guest@avgismatullin:/home/guest
[guest@avgismatullin ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 24488 anp 13 18:40 simpleid2
[guest@avgismatullin ~]$ ./simpleid2
euid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@avgismatullin ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@avgismatullin ~]$ su
Пароль:
[root@avgismatullin guest]# chmod g+s simpleid2
[root@avgismatullin guest]# ./simpleid2
euid=0, e_gid=1001
real_uid=0, real_gid=0
[root@avgismatullin guest]#
```

Рис. 3.5: Командная строка. Сравнение результатов вывода

1. После этого пропишем эту программу (рис. 3.6)

```
guest@avgismatullin:~ — nano readfile.c
GNU nano 5.6.1 readfile.c
#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 (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. 3.6: Редактор. Файл readfile

После компиляции сменим владельца у файла readfile.c и поменяем права так, чтобы только root пользователь смог его прочитать, установим SetU'D-бит и, наконец, проверим, может ли программа readfile прочитать файл readfile.c (рис. 3.7)

A terminal window titled 'guest@avgismatullin:~' with a search icon in the top right. The terminal shows a series of commands executed as root: 'chown root:root readfile', 'chmod o-r readfile.c', 'chmod g-rw readfile.c', 'chmod u+s readfile', and 'exit'. Then, the user 'guest' runs 'cat readfile.c'. The output shows the beginning of a C program with headers <fcntl.h>, <stdio.h>, <sys/stat.h>, <sys/types.h>, and <unistd.h>. The code defines an 'int' and a 'main' function that takes 'argc' and 'argv'. Inside 'main', it declares 'unsigned char buffer[16]', 'size_t bytes_read', and 'int i'. It then opens a file with 'open(argv[1], O_RDONLY);'.

```
guest@avgismatullin:~  
[root@avgismatullin guest]# chown root:root readfile  
[root@avgismatullin guest]# chmod o-r readfile.c  
[root@avgismatullin guest]# chmod g-rw readfile.c  
[root@avgismatullin guest]# chmod u+s readfile  
[root@avgismatullin guest]# exit  
exit  
[guest@avgismatullin ~]$ cat 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);
```

Рис. 3.7: Командная строка. Проверка изменений

3.2 Исследование Sticky-бита

1. Начнем выполнение с того, что узнаем, есть ли атрибут Sticky на директории /tmp. От имени пользователя guest запишем в файл file01.txt слово test, а затем разрешим для остальных пользователей читать и записывать этот файл. Заметим, что ни дописать, ни перезаписать, ни удалить этот файл мы с пользователем guest2 не можем (рис. 3.8)

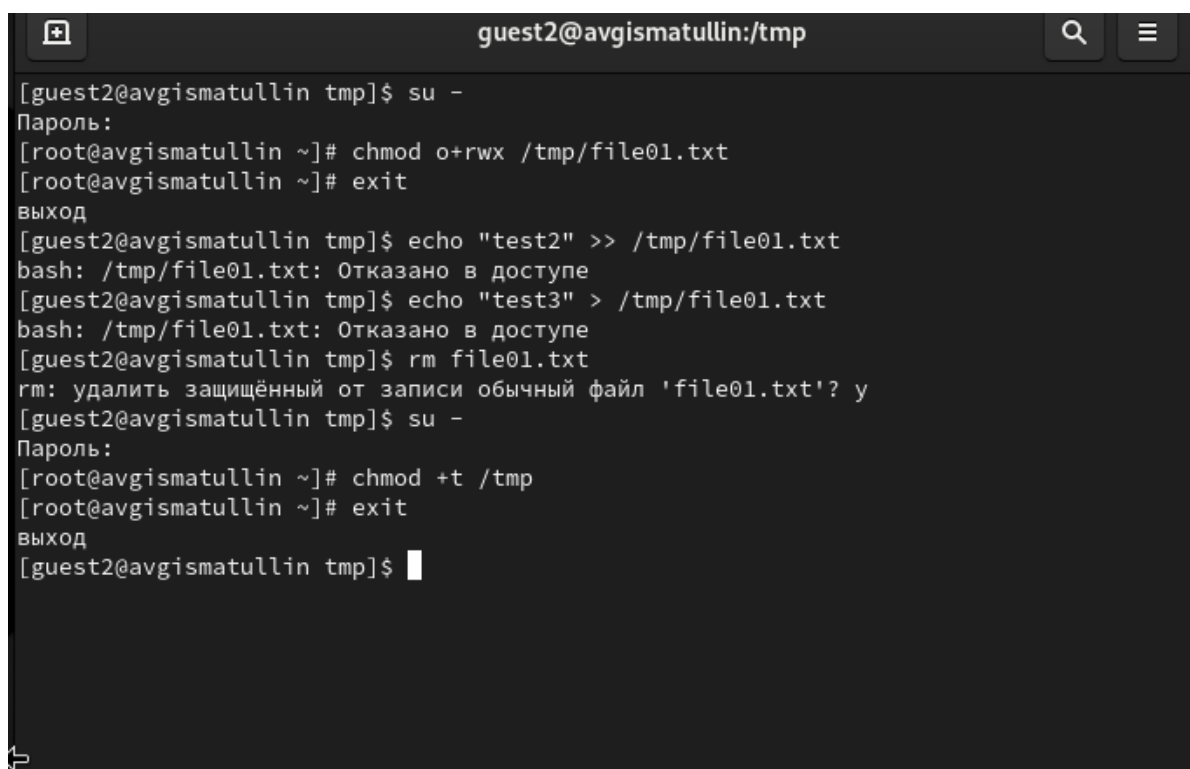
```

[guest@avgismatullin ~]$ ls -l / | grep tmp
drwxrwxrwt. 15 root root 4096 anp 13 19:02 tmp
[guest@avgismatullin ~]$ echo "test" > /tmp/file01.txt
[guest@avgismatullin ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 anp 13 19:06 /tmp/file01.txt
[guest@avgismatullin ~]$ chmod o_rw /tmp/file01.txt
chmod: неверный режим: «o_rw»
По команде «chmod --help» можно получить дополнительную информацию.
[guest@avgismatullin ~]$ chmod o+rw /tmp/file01.txt
[guest@avgismatullin ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 anp 13 19:06 /tmp/file01.txt
[guest@avgismatullin ~]$ su guest2
Пароль:
[guest2@avgismatullin guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@avgismatullin guest]$ cat /tmp/file01.txt
test
[guest2@avgismatullin guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@avgismatullin guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@avgismatullin guest]$ cat /tmp/file01.txt
test
[guest2@avgismatullin guest]$

```

Рис. 3.8: Командная строка. Исследование Sticky-бита

2. Изменим расширенный атрибут `t` в директории `/tmp` и убедимся, что теперь мы можем удалить файл `file01.txt` (рис. 3.9)



```
guest2@avgismatullin:/tmp
[guest2@avgismatullin tmp]$ su -
Пароль:
[root@avgismatullin ~]# chmod o+rwX /tmp/file01.txt
[root@avgismatullin ~]# exit
выход
[guest2@avgismatullin tmp]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@avgismatullin tmp]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@avgismatullin tmp]$ rm file01.txt
rm: удалить защищённый от записи обычный файл 'file01.txt'? y
[guest2@avgismatullin tmp]$ su -
Пароль:
[root@avgismatullin ~]# chmod +t /tmp
[root@avgismatullin ~]# exit
выход
[guest2@avgismatullin tmp]$
```

Рис. 3.9: Командная строка. Исследование Sticky-бита

4 Выводы

В ходе выполнения данной лабораторной работы были изучены механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получены практические навыки работы в консоли с дополнительными атрибутами. Рассмотрены работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов

Список литературы

1. Теория разграничения прав пользователей
2. Разрешения доступа к файлам