

ТЕМА « Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов »

Выполнил:

Студент группы НПИбд-02-21

Студенческий билет № 1032205641

Сатлихана Петрити

Введение

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов.

Получение практических навыков работы в консоли с дополнительными атрибутами.

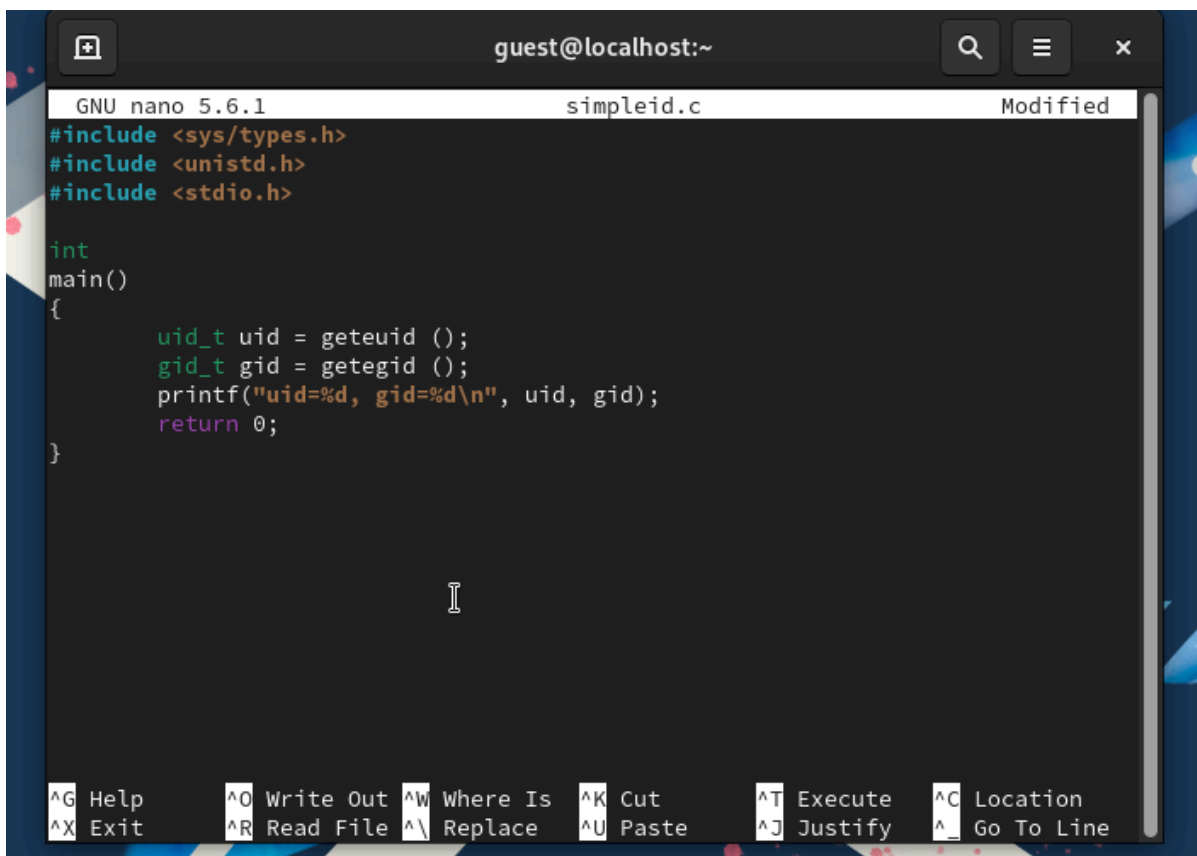
Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Последовательность выполнения работы

5.3.1. Создание программы

Войдите в систему от имени пользователя guest.

Создайте программу simpleid.c:



```
GNU nano 5.6.1 simpleid.c Modified
#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. 1 Создание программы simpleid.c

Скомпилируйте программу и убедитесь, что файл программы создан:

```
gcc simpleid.c -o simpleid
```

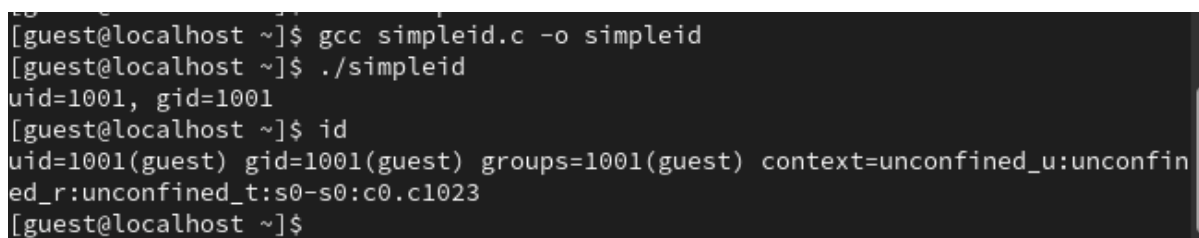
Выполните программу simpleid:

```
./simpleid
```

Выполните системную программу id:

id и сравните полученный вами результат с данными предыдущего пункта задания.

Один и тот же uid и gid

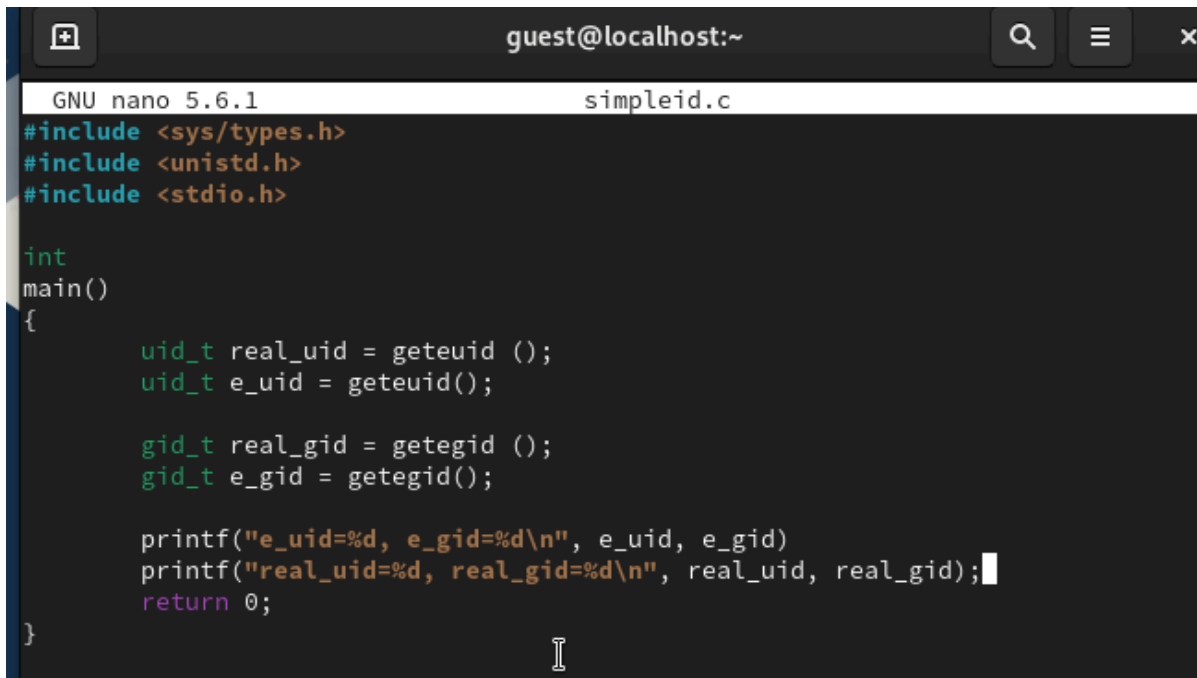


```
[guest@localhost ~]$ gcc simpleid.c -o simpleid
[guest@localhost ~]$ ./simpleid
uid=1001, gid=1001
[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost ~]$
```

рис. 2 Пункты 3,4,5: компиляция и id

Усложните программу, добавив вывод действительных идентификаторов:

Получившуюся программу назовите simpleid2.c.



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

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

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

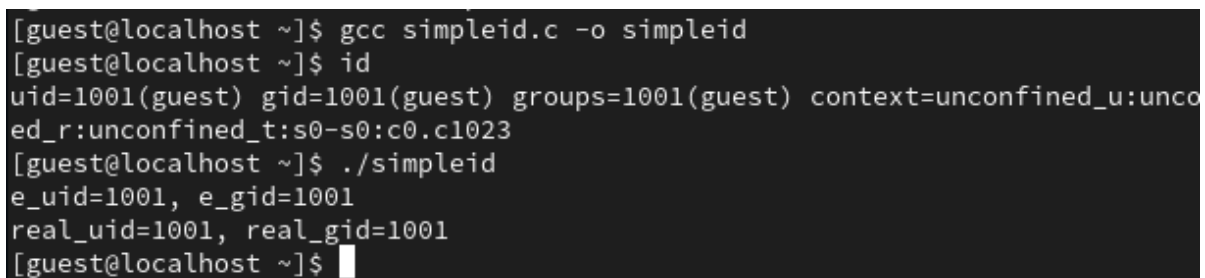
    printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid)
    printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
    return 0;
}
```

рис. 3 Создание программы simpleid.c(вместо simpleid2.c)

Скомпилируйте и запустите simpleid2.c:

```
gcc simpleid2.c -o simpleid2
```

```
./simpleid2
```



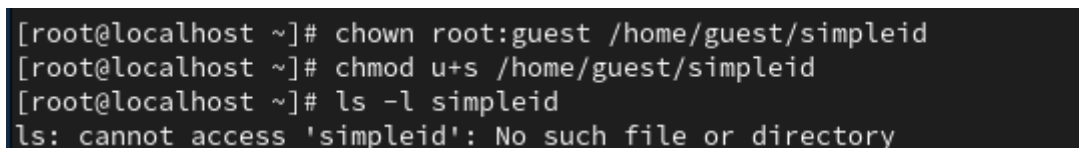
```
[guest@localhost ~]$ gcc simpleid.c -o simpleid
[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unco
ed_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost ~]$ ./simpleid
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@localhost ~]$
```

рис. 4 Компиляция и запуск simpleid2.c

От имени суперпользователя выполните команды:

```
chown root:guest /home/guest/simpleid2
```

```
chmod u+s /home/guest/simpleid2
```



```
[root@localhost ~]# chown root:guest /home/guest/simpleid
[root@localhost ~]# chmod u+s /home/guest/simpleid
[root@localhost ~]# ls -l simpleid
ls: cannot access 'simpleid': No such file or directory
```

рис. 5 выполнение команд chown,chmod

Используйте sudo или повысьте временно свои права с помощью su.

Поясните, что делают эти команды.

sudo (SuperUser DO)

- **Что это?** `sudo` — это команда, которая позволяет пользователям выполнять команды с привилегиями суперпользователя (root) или другого пользователя, указанного в конфигурационном файле `/etc/sudoers`.

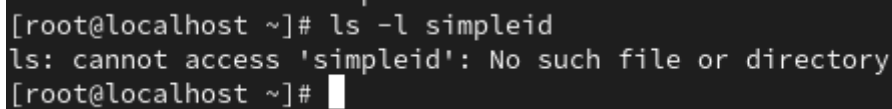
su (Substitute User)

- **Что это?** `su` — это команда, которая позволяет пользователю переключиться на другого пользователя, обычно на суперпользователя (root).

Выполните проверку правильности установки новых атрибутов и смены

владельца файла `simpleid2`:

`ls -l simpleid2`



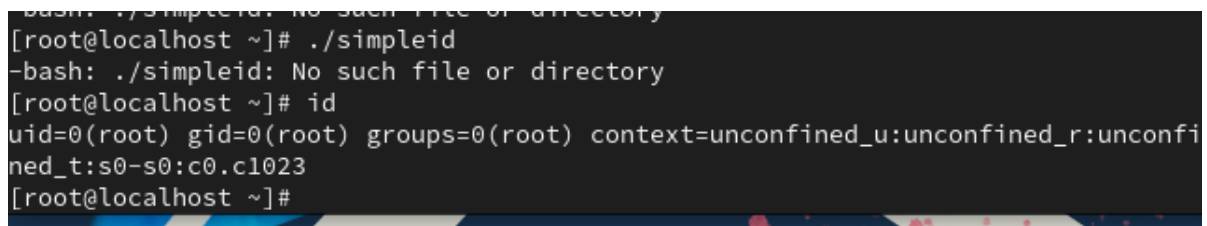
```
[root@localhost ~]# ls -l simpleid
ls: cannot access 'simpleid': No such file or directory
[root@localhost ~]#
```

рис. 6 проверка правильности установки новых атрибутов

Запустите `simpleid2` и `id`:

`./simpleid2`

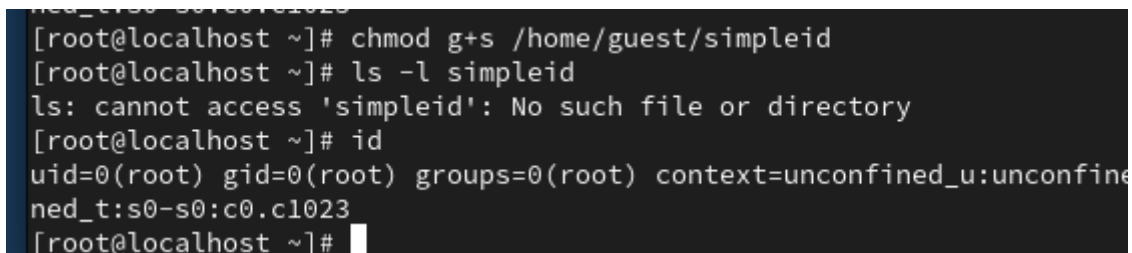
`id` Сравните результаты.



```
[root@localhost ~]# ./simpleid
-bash: ./simpleid: No such file or directory
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

рис. 7 запуск и `id`

Проделайте тоже самое относительно SetGID-бита.



```
[root@localhost ~]# chmod g+s /home/guest/simpleid
[root@localhost ~]# ls -l simpleid
ls: cannot access 'simpleid': No such file or directory
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

рис. 8 SetGID-бита.

Создайте программу `readfile.c`:

```
GNU nano 5.6.1                                readfile.c                                Mod
#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));
}
```

рис. 9 Создание программы readfile.c

Откомпилируйте её.

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

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.

Проверьте, что пользователь guest не может прочитать файл readfile.c.

```

[root@localhost ~]# gcc readfile.c -o readfile
[root@localhost ~]# nano readfile.c
[root@localhost ~]# chown root:root readfile.c
[root@localhost ~]# chmod 400 readfile.c
[root@localhost ~]# 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);
    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;
}
[root@localhost ~]# exit
logout
[guest@localhost ~]$ cat readfile.c
cat: readfile.c: No such file or directory
[guest@localhost ~]$ █

```

рис. 10 Пункты 14,15,16

Смените у программы readfile владельца и установите SetU'D-бит.

Проверьте, может ли программа readfile прочитать файл readfile.c? Да, читается.

Проверьте, может ли программа readfile прочитать файл /etc/shadow? Да, читается.

```

[root@localhost ~]# chown root:root readfile
[root@localhost ~]# chmod u+s readfile
[root@localhost ~]# ./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;
}
[root@localhost ~]# ./readfile /etc/shadow
root:$6$sMnGGfarD0kpTmFY$Bjf/8CuoHdRQZbYzI1lyn.N0BgpIYJ/2Lbqb25zTn4g60dZeu6JSDPCMJqc3Abk08LFgRknshzg
995:0:99999:7:::
bin:!:19820:0:99999:7:::
daemon:!:19820:0:99999:7:::

```

рис. 11 Пункты 17,18,19

5.3.2. Исследование Sticky*-бума**

Выясните, установлен ли атрибут Sticky на директории /tmp, для чего

выполните команду

```
ls -l / | grep tmp
```

От имени пользователя guest создайте файл file01.txt в директории /tmp

со словом test:

```
echo "test" > /tmp/file01.txt
```

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
```

```

[root@localhost ~]# ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 Sep 29 16:51 tmp
[root@localhost ~]# echo "test" > /tmp/file01.txt
[root@localhost ~]# ls -l /tmp/file01.txt
-rw-r--r--. 1 root root 5 Sep 29 16:57 /tmp/file01.txt
[root@localhost ~]# chmod o+rw /tmp/file01.txt
[root@localhost ~]# ls -l /tmp/file01.txt
-rw-r--rw-. 1 root root 5 Sep 29 16:57 /tmp/file01.txt

```

рис. 12 Пункты 1,2,3

От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt:

```
cat /tmp/file01.txt
```

От пользователя guest2 попробуйте дозаписать в файл

/tmp/file01.txt слово test2 командой `echo "test2" > /tmp/file01.txt` Удалось ли вам выполнить операцию?

Проверьте содержимое файла командой

```
cat /tmp/file01.txt
```

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
```

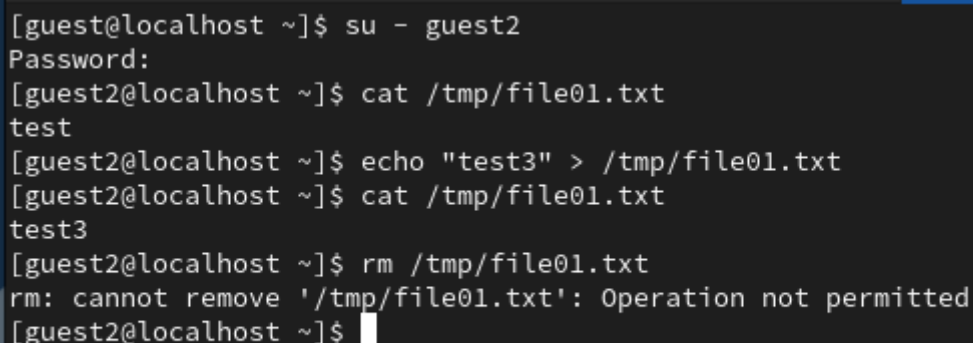
Удалось ли вам выполнить операцию?

Проверьте содержимое файла командой

```
cat /tmp/file01.txt
```

От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой

`rm /tmp/file01.txt` Удалось ли вам удалить файл? Нет, операция была запрещена.



```
[guest@localhost ~]$ su - guest2
Password:
[guest2@localhost ~]$ cat /tmp/file01.txt
test
[guest2@localhost ~]$ echo "test3" > /tmp/file01.txt
[guest2@localhost ~]$ cat /tmp/file01.txt
test3
[guest2@localhost ~]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@localhost ~]$
```

рис. 13 Пункты 4,5,6,7,8

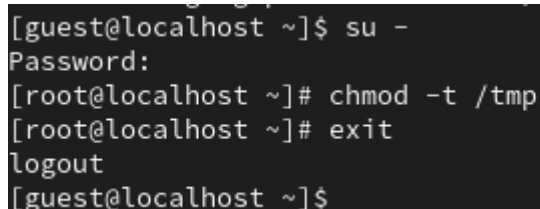
Повысьте свои права до суперпользователя следующей командой

`su -` и выполните после этого команду, снимающую атрибут `t` (Sticky-бит) с директории /tmp:

```
chmod -t /tmp
```

Покиньте режим суперпользователя командой

`Exit`



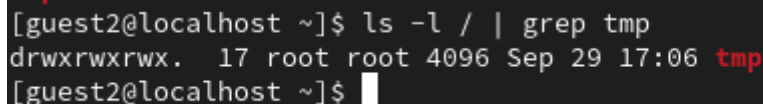
```
[guest@localhost ~]$ su -
Password:
[root@localhost ~]# chmod -t /tmp
[root@localhost ~]# exit
logout
[guest@localhost ~]$
```

рис. 14 Пункты 10,11

От пользователя guest2 проверьте, что атрибута t у директории /tmp

нет:

ls -l / | grep tmp



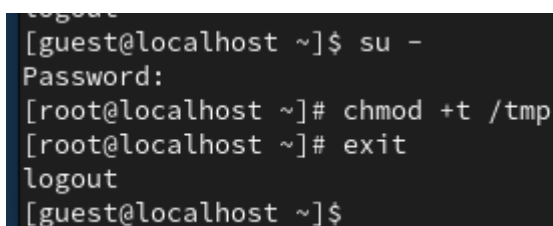
```
[guest2@localhost ~]$ ls -l / | grep tmp
drwxrwxrwx. 17 root root 4096 Sep 29 17:06 tmp
[guest2@localhost ~]$
```

рис. 15 Нет атрибута t у директории /tmp

Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp:

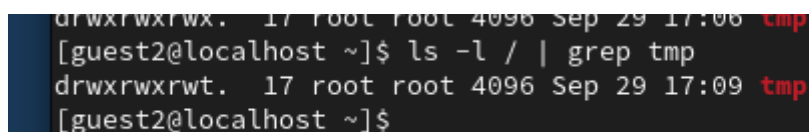
su - chmod +t /tmp

exit



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

рис. 16 Возвращающий атрибут t



```
drwxrwxrwt. 17 root root 4096 Sep 29 17:09 tmp
[guest2@localhost ~]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 Sep 29 17:09 tmp
[guest2@localhost ~]$
```

рис. 17 Проверка

Вывод

В этой лабораторной работе мы изучили, как работают специальные атрибуты файлов в Linux, такие как SetUID, SetGID и Sticky-бит. Мы увидели, как они помогают контролировать доступ к файлам и программам, а также защищают общие папки от удаления файлов другими пользователями. Эти механизмы важны для безопасности и защиты данных в системах с несколькими пользователями.