Лабораторная работы №5

Павлова В.Ю.

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.

Ход работы

Войдите в систему от имени пользователя guest.

Создайте программу simpleid.c:: (рис. [-@fig:001]) (рис. [-@fig:002])

touch

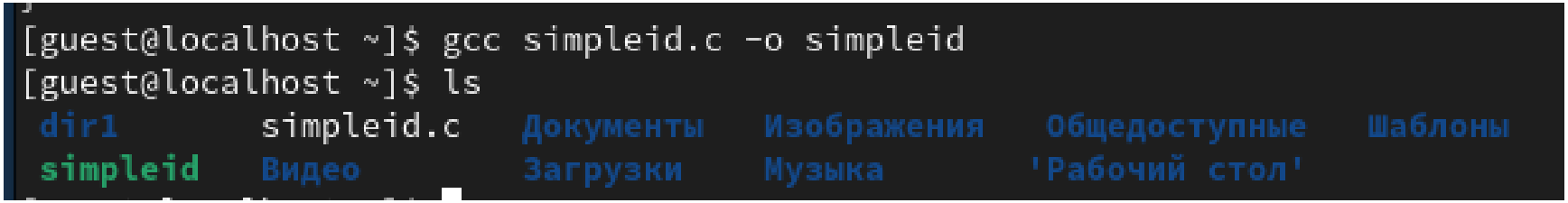
```
[guest@localhost ~]$ nano simpleid.c
[guest@localhost ~]$ cat simpleid.c
#include  <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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

simpleid.c

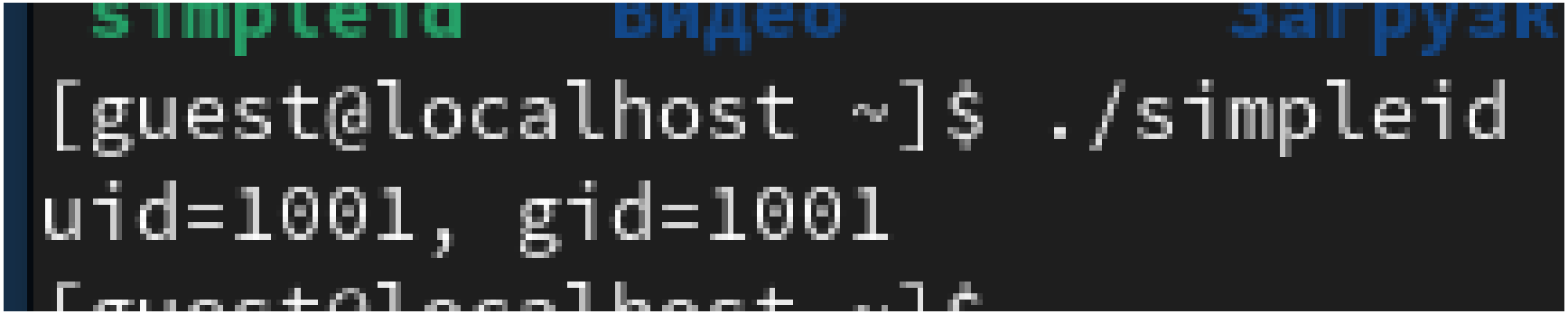
Ход работы

комплилируйте программу и убедитесь, что файл программы создан: (рис. [-@fig:003])


[guest@localhost ~]$ gcc simpleid.c -o simpleid
[guest@localhost ~]$ ls
dir1 simpleid.c Документы Изображения Общедоступные Шаблоны
simpleid Видео Загрузки Музыка 'Рабочий стол'

компиляция

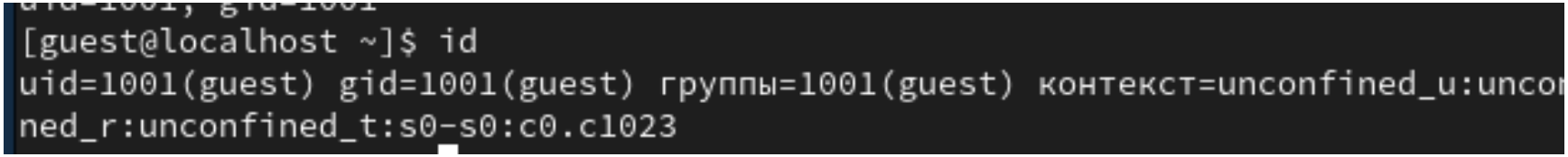
Выполните программу simpleid: (рис. [-@fig:004])


simpleid
Видео
Загрузк
[guest@localhost ~]$ ./simpleid
uid=1001, gid=1001

выполнение

Ход работы

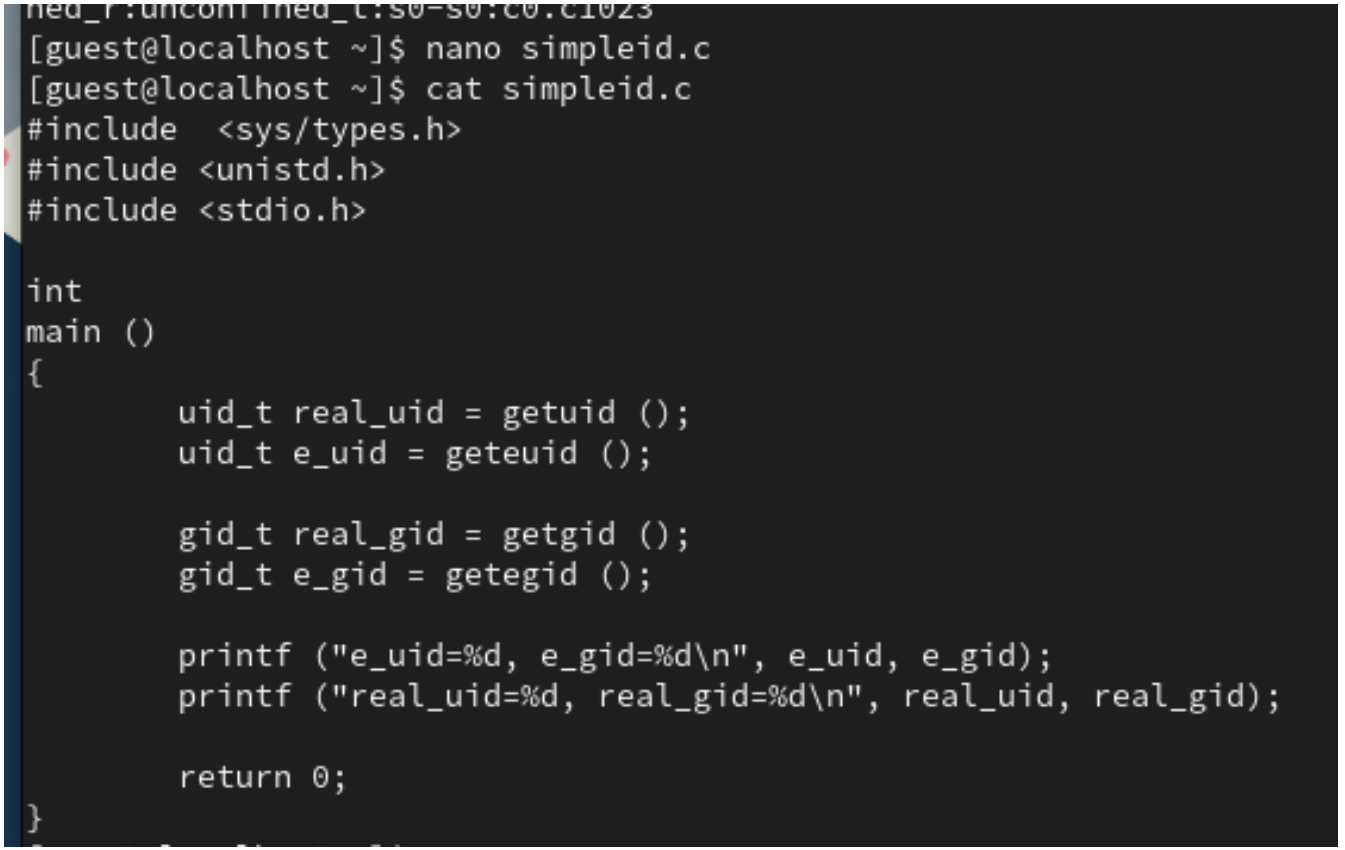
Выполните системную программу id: (рис. [-@fig:005])


[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unco
ned_r:unconfined_t:s0-s0:c0.c1023

id

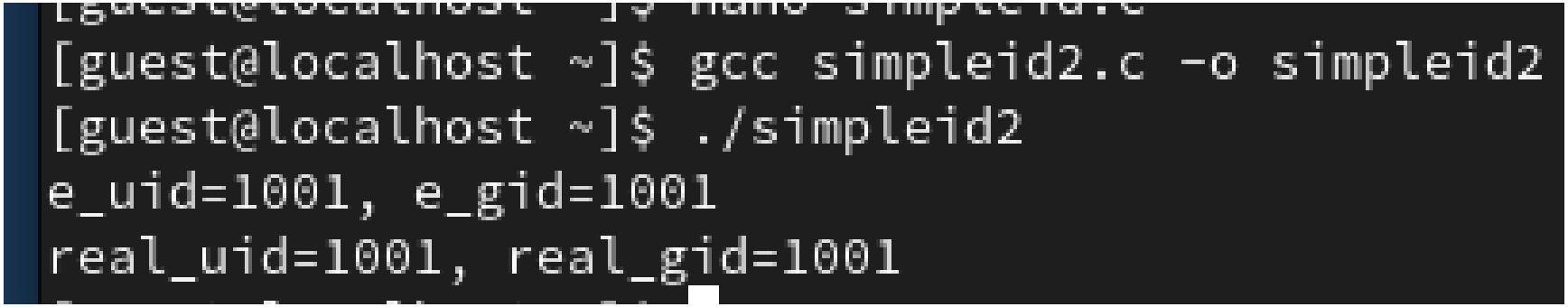
Ход работы

Усложните программу, добавив вывод действительных идентификаторов: (рис. [-@fig:006])


ned_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost ~]$ nano simpleid.c
[guest@localhost ~]$ cat simpleid.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;
}

simpleid2

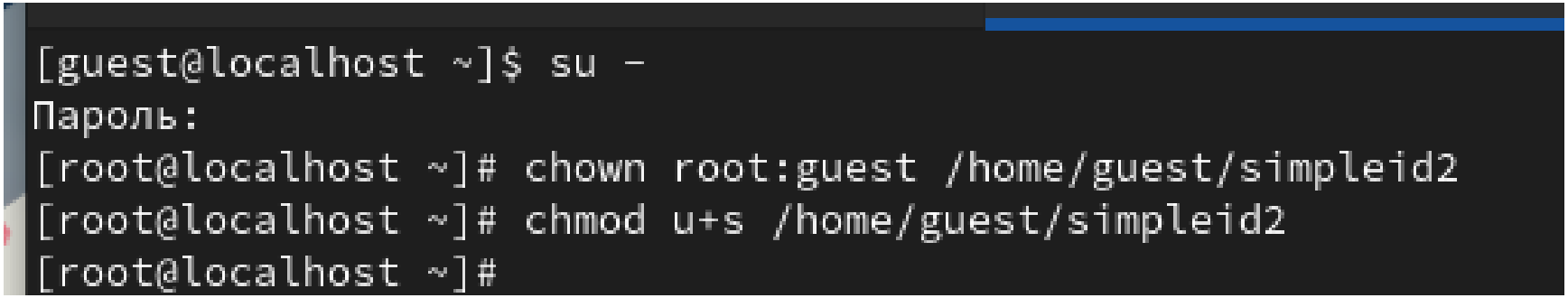
Скомпилируйте и запустите simpleid2.c: (рис. [-@fig:007])


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

компиляция

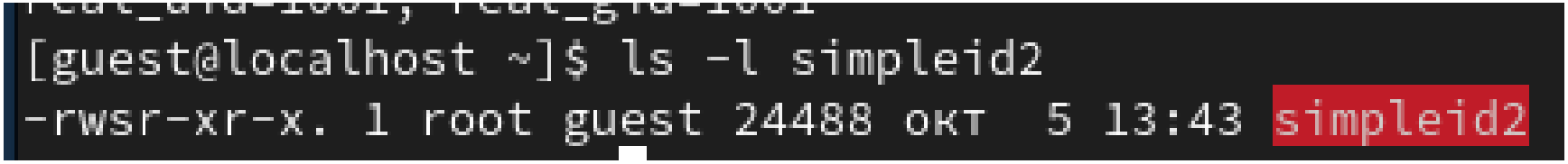
Ход работы

От имени суперпользователя выполните команды: (рис. [-@fig:008])


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

команды

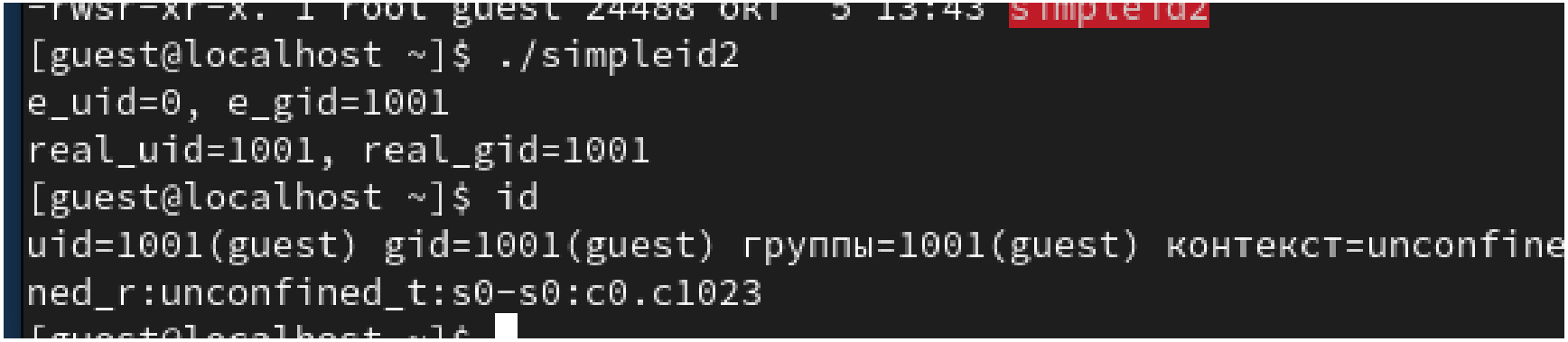
Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2: (рис. [-@fig:009])


[guest@localhost ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 24488 окт 5 13:43 simpleid2

ls -l

Ход работы

Запустите simpleid2 и id: (рис. [-@fig:010])


-rwsr-xr-x. 1 root guest 24488 окт 5 13:43 simpleid2
[guest@localhost ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfine
ned_r:unconfined_t:s0-s0:c0.c1023

запуск

{ #fig:010 width=70%

Ход работы

Создайте программу readfile.c:(рис. [-@fig:011])

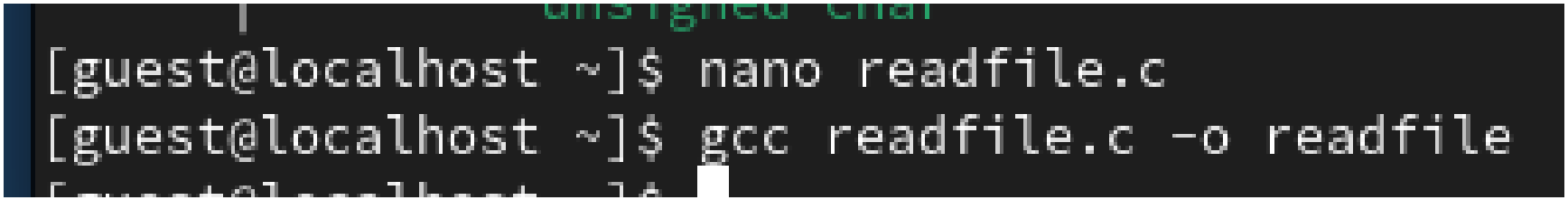
```
[guest@localhost ~]$ cat readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

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;
}
```

readfile.c

Откомпилируйте её. (рис. [-@fig:012])


[guest@localhost ~]$ nano readfile.c
[guest@localhost ~]$ gcc readfile.c -o readfile

компиляция

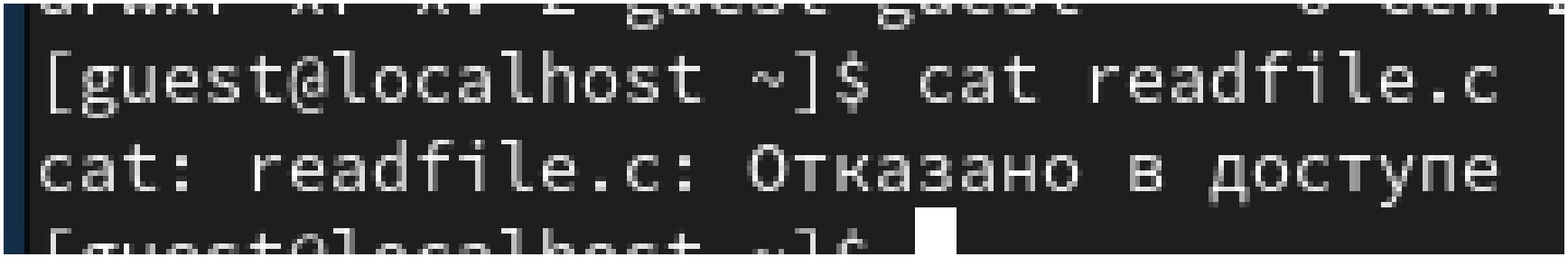
Ход работы

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. (рис. [-@fig:013])

```
[root@localhost ~]# chown root:root /home/guest/readfile.c
[root@localhost ~]# chmod o-rwx /home/guest/readfile.c
```

chmod

Проверьте, что пользователь guest не может прочитать файл readfile.c.(рис. [-@fig:014])


[guest@localhost ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе

cat

Ход работы

Смените у программы readfile владельца и установите SetU’D-бит. (рис. [-@fig:015])

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

chmod

Ход работы

Проверьте, может ли программа readfile прочитать файл readfile.c? (рис. [-@fig:016])

```
cat: readfile.c: Отказано в доступе
[guest@localhost ~]$ ./readfile readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
```

readfile

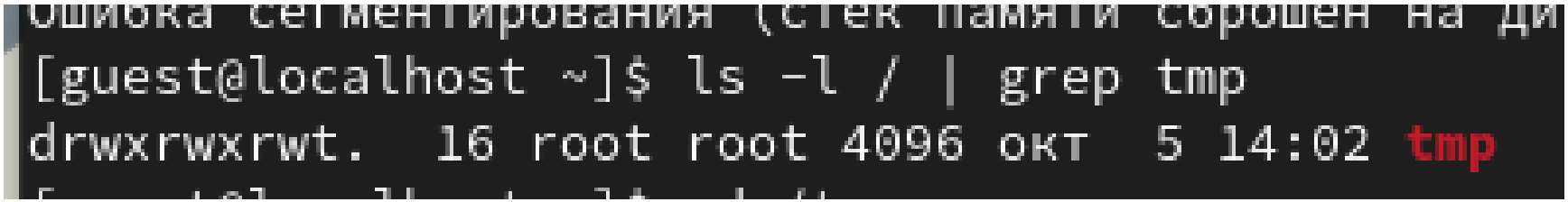
Проверьте, может ли программа readfile прочитать файл /etc/shadow? (рис. [-@fig:017])


[guest@localhost ~]$ ./readfile etc/shadow
86_64./readfileetc/shadowSHELL=/bin/bashSESSION_M

readfile

Ход работы

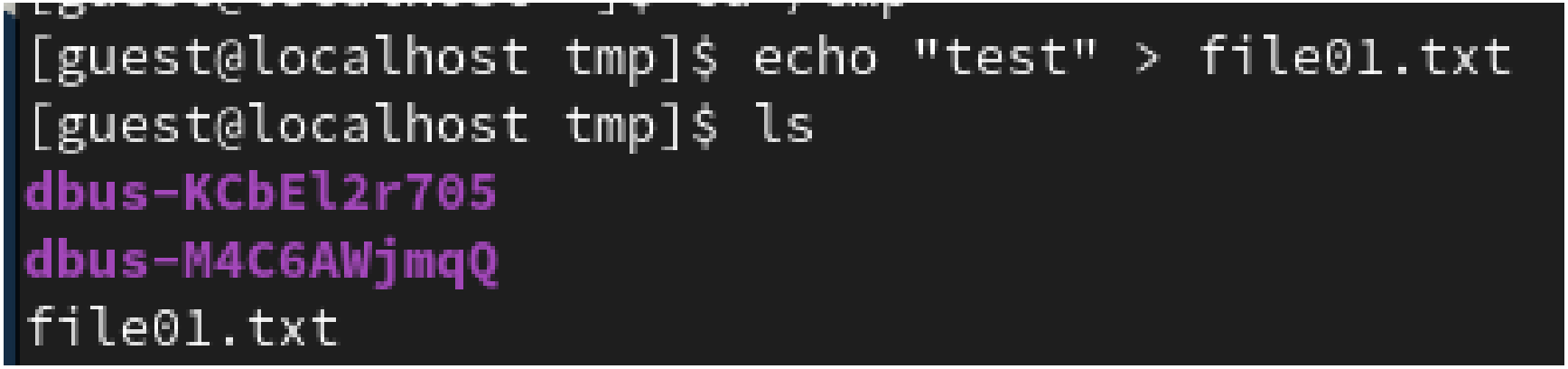
Выясните, установлен ли атрибут Sticky на директории /tmp (рис. [-@fig:018])


Ошибка сегментирования (стек памяти сброшен на ди
[guest@localhost ~]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 окт 5 14:02 tmp

ls

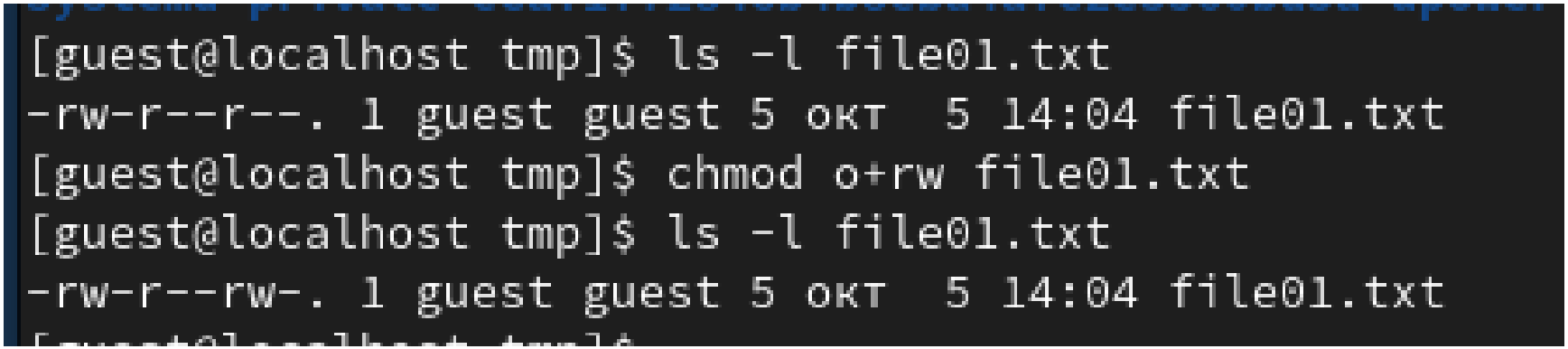
Ход работы

От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: (рис. [-@fig:019])


[guest@localhost tmp]$ echo "test" > file01.txt
[guest@localhost tmp]$ ls
dbus-KCbEl2r705
dbus-M4C6AWjmqQ
file01.txt

echo

Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные»: (рис. [-@fig:020])


[guest@localhost tmp]$ ls -l file01.txt
-rw-r--r--. 1 guest guest 5 окт 5 14:04 file01.txt
[guest@localhost tmp]$ chmod o+rw file01.txt
[guest@localhost tmp]$ ls -l file01.txt
-rw-r--rw-. 1 guest guest 5 окт 5 14:04 file01.txt

chmod

Ход работы

От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt:(рис. [-@fig:021])

cat

От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 (рис. [-@fig:022])

```
[guest2@localhost tmp]$ echo "test2" > file01.txt
```

echo

Проверьте содержимое файла командой(рис. [-@fig:023])

cat

Ход работы

От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3(рис. [-@fig:024])

chmod

Проверьте содержимое файла командой(рис. [-@fig:025])

cat

Ход работы

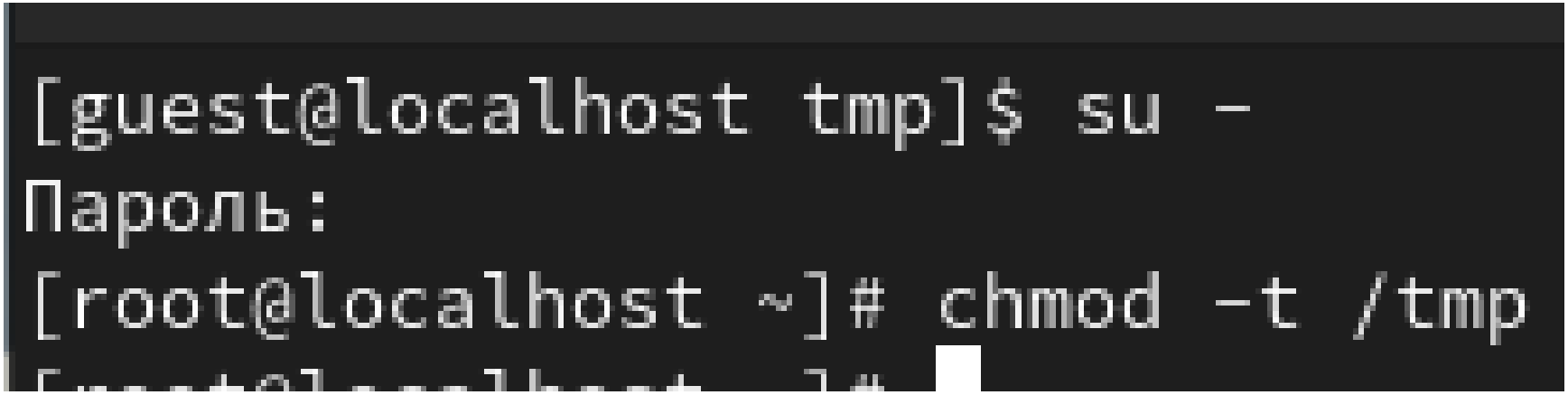
От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой (рис. [-@fig:026])


[guest2@localhost tmp]$ rm file01.txt
rm: невозможно удалить 'file01.txt': Операция не позволена

rm

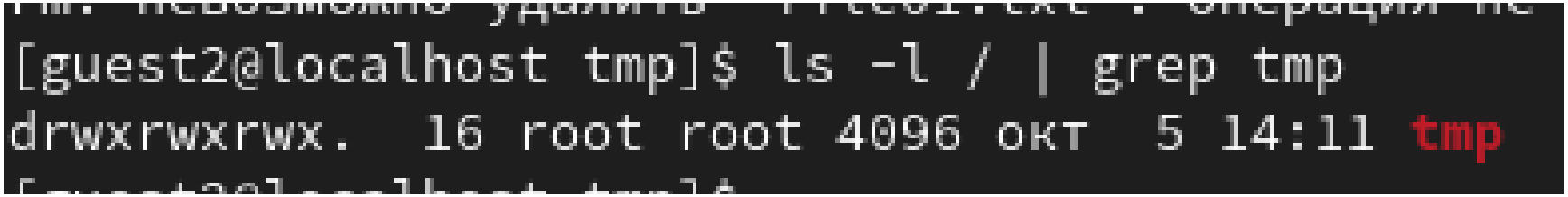
Ход работы

Повысьте свои права до суперпользователя и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp:(рис. [-@fig:027])



chmod

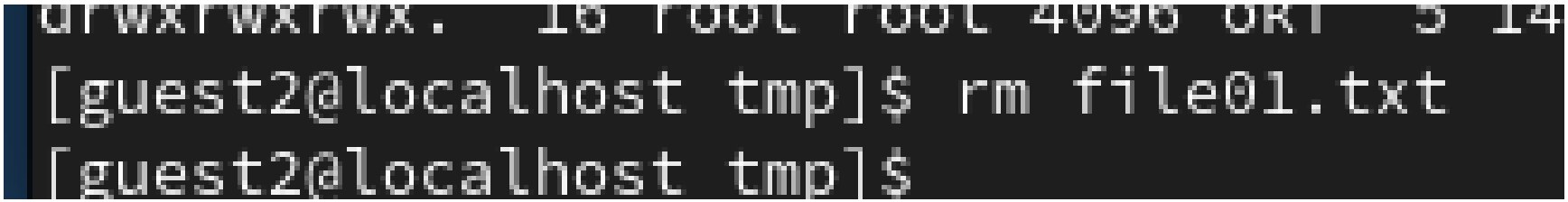
От пользователя guest2 проверьте, что атрибута t у директории /tmp нет: (рис. [-@fig:028])


[guest2@localhost tmp]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 окт 5 14:11 tmp

ls

Ход работы

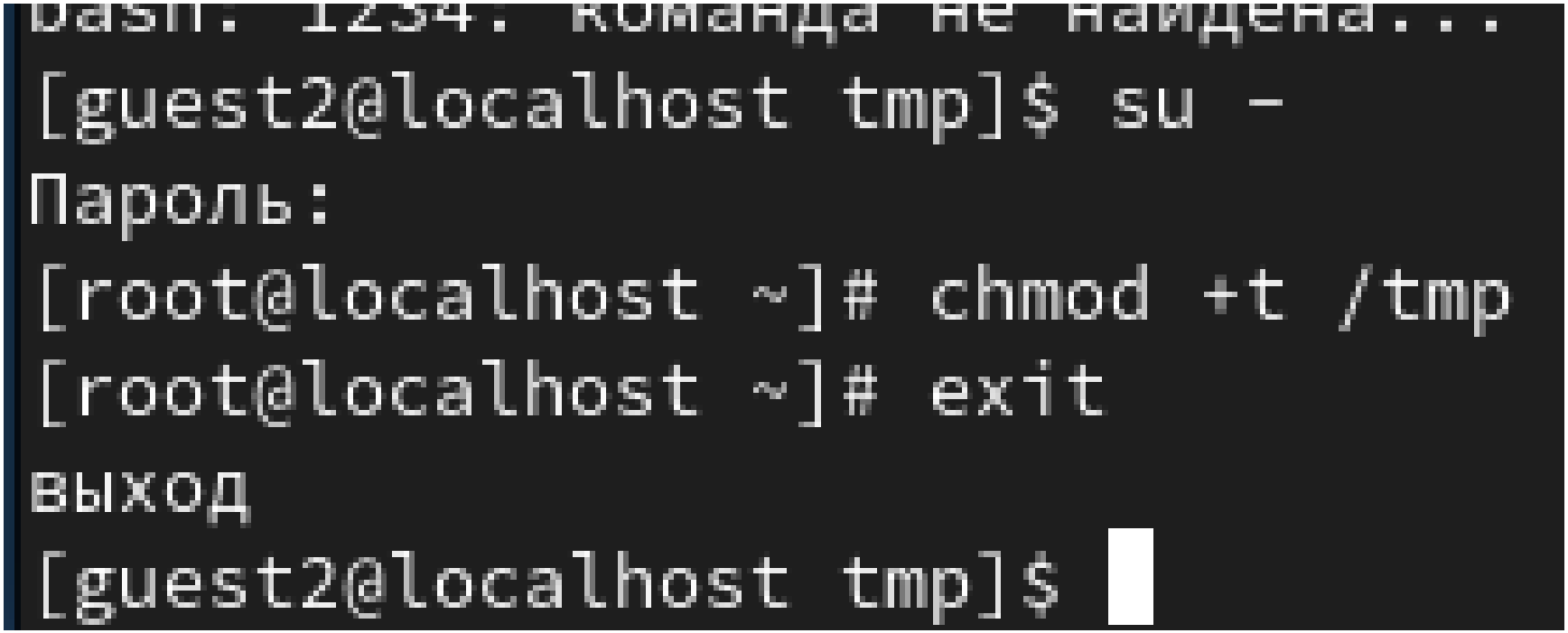
Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем?(рис. [-@fig:029])


drwxrwxrwx. 18 root root 4096 окт 5 14
[guest2@localhost tmp]$ rm file01.txt
[guest2@localhost tmp]$

rm

Ход работы

Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp: (рис. [-@fig:030])


bash: 1234: команда не найдена...
[guest2@localhost tmp]$ su -
Пароль:
[root@localhost ~]# chmod +t /tmp
[root@localhost ~]# exit
выход
[guest2@localhost tmp]$

chmod

Вывод

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