

Лабораторная работа №5

Информационная безопасность

Александрова Ульяна Вадимовна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Подготовка к выполнению работы	7
4	Выполнение лабораторной работы	9
4.1	Создание программы	9
4.2	Исследование Sticky-бита	14
5	Выводы	16

Список иллюстраций

3.1	Проверка установки ПО	7
3.2	setenforce 0	8
4.1	Вход в систему от другого пользователя	9
4.2	Создание программы	9
4.3	Заполнение элементарной программы	10
4.4	Компиляция и запуск программы	10
4.5	Команда id	10
4.6	Заполнение программы	11
4.7	Компиляция и запуск программы	11
4.8	Поменяла владельца программы	11
4.9	ls -l	12
4.10	Сравнение результатов	12
4.11	Создание программы	12
4.12	Компиляция программы	13
4.13	Смена владельца	13
4.14	Результат работы программы	13
4.15	Выполнение задач	14
4.16	Работа с файлами	14
4.17	Работа с атрибутом t	15
4.18	Возвращение Sticky	15

Список таблиц

1 Цель работы

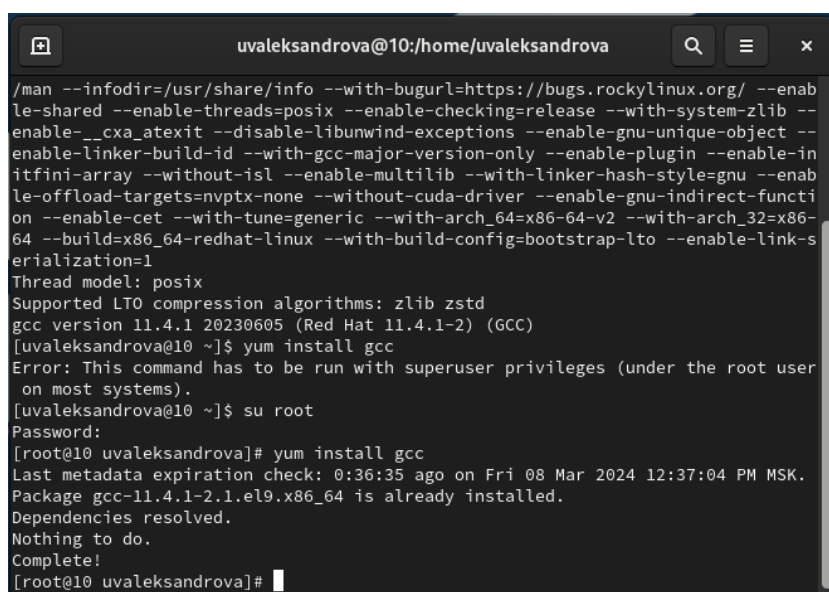
Целью работы является изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в консоли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

2 Теоретическое введение

Setuid, Setgid и Sticky Bit - это специальные типы разрешений позволяют задавать расширенные права доступа на файлы или каталоги.

3 Подготовка к выполнению работы

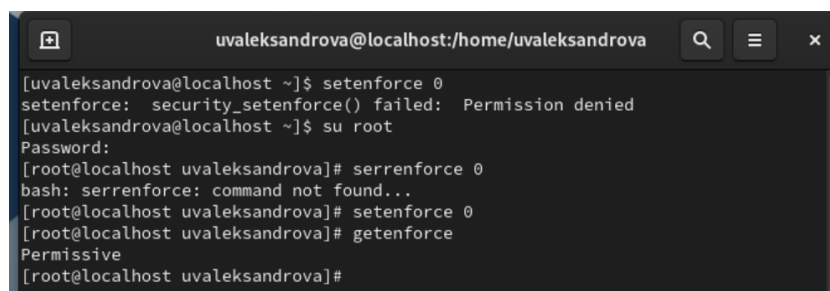
Я проверила, установлен ли у меня gcc командой **yum install gcc**. Он установлен и обновлен до последней версии (рис. 3.1).



```
uvaleksandrova@10:/home/uvaleksandrova
/man --infodir=/usr/share/info --with-bugurl=https://bugs.rockylinux.org/ --enable-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-initfini-array --without-isl --enable-multilib --with-linker-hash-style=gnu --enable-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function --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-serialization=1
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)
[uvaleksandrova@10 ~]$ yum install gcc
Error: This command has to be run with superuser privileges (under the root user on most systems).
[uvaleksandrova@10 ~]$ su root
Password:
[root@10 uvaleksandrova]# yum install gcc
Last metadata expiration check: 0:36:35 ago on Fri 08 Mar 2024 12:37:04 PM MSK.
Package gcc-11.4.1-2.1.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@10 uvaleksandrova]#
```

Рис. 3.1: Проверка установки ПО

Помимо этого, я отключила систему запретов до очередной перезагрузки системы командой **setenforce 0**. После этого команда **getenforce** выводит **Permissive** (рис. 3.2).



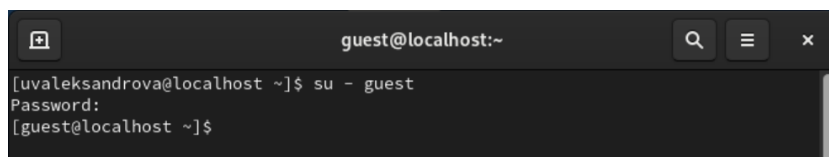
```
uvaleksandrova@localhost: /home/uvaleksandrova
[uvaleksandrova@localhost ~]$ setenforce 0
setenforce: security_setenforce() failed: Permission denied
[uvaleksandrova@localhost ~]$ su root
Password:
[root@localhost uvaleksandrova]# serrenforce 0
bash: serrenforce: command not found...
[root@localhost uvaleksandrova]# setenforce 0
[root@localhost uvaleksandrova]# getenforce
Permissive
[root@localhost uvaleksandrova]#
```

Рис. 3.2: setenforce 0

4 Выполнение лабораторной работы

4.1 Создание программы

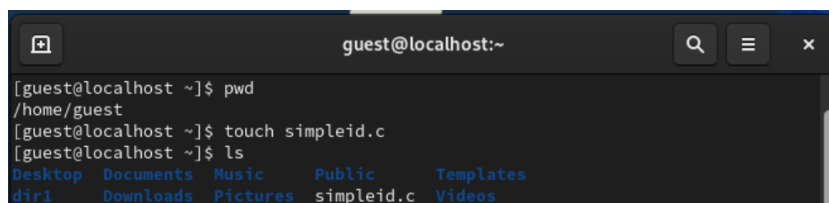
Я вошла в систему от имени пользователя guest (рис. 4.1).



```
guest@localhost:~  
[uvaleksandrova@localhost ~]$ su - guest  
Password:  
[guest@localhost ~]$
```

Рис. 4.1: Вход в систему от другого пользователя

Далее создала программу simpleid.c и заполнила ее (рис. 4.2), (рис. 4.3).



```
guest@localhost:~  
[guest@localhost ~]$ pwd  
/home/guest  
[guest@localhost ~]$ touch simpleid.c  
[guest@localhost ~]$ ls  
Desktop  Documents  Music      Public      Templates  
dir1     Downloads  Pictures   simpleid.c  Videos
```

Рис. 4.2: Создание программы

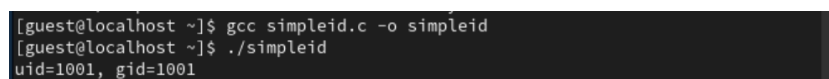


```
GNU nano 5.6.1 simpleid.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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

Рис. 4.3: Заполнение элементарной программы

Скомпилировала файл через **gcc simpleid.c -o simpleid** и выполнила программу simpleid (рис. 4.4).



```
[guest@localhost ~]$ gcc simpleid.c -o simpleid
[guest@localhost ~]$ ./simpleid
uid=1001, gid=1001
```

Рис. 4.4: Компиляция и запуск программы

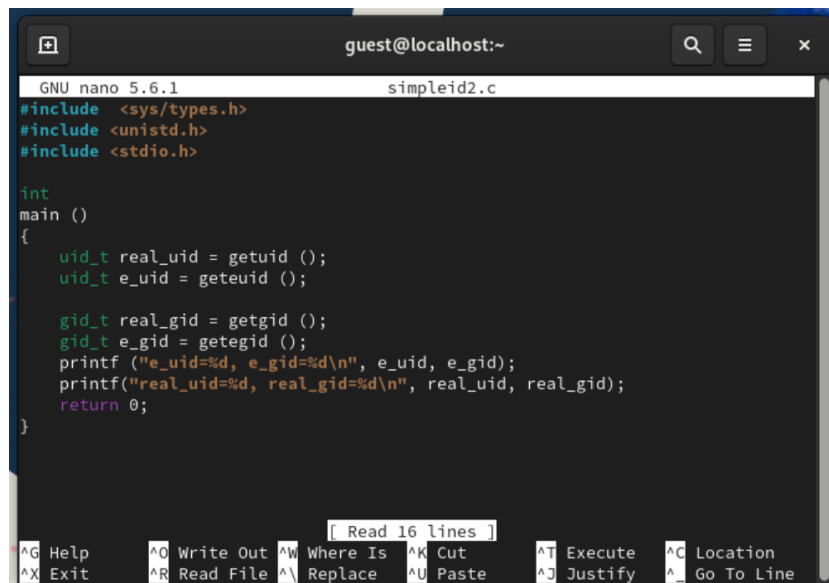
Выполнила системную программу id. Результаты похожи. Gid и uid одинаковые, однако команда id дает больше информации (рис. 4.5).



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

Рис. 4.5: Команда id

Усложнила программу, добавив вывод действительных идентификаторов (рис. 4.6).



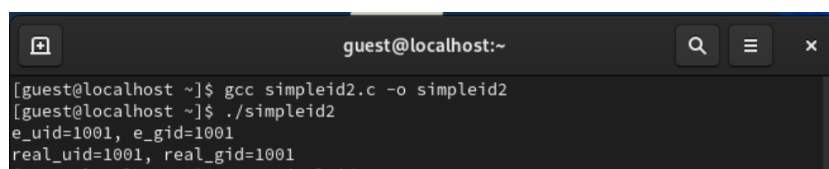
```
GNU nano 5.6.1 simpleid2.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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: Заполнение программы

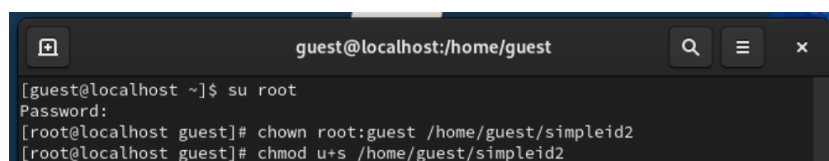
Скомпилировала и запустила simpleid2.c (рис. 4.7).



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

Рис. 4.7: Компиляция и запуск программы

От имени суперпользователя выполнила команды **chown root:guest /home/guest/simpleid2**, **chmod u+s /home/guest/simpleid2** (рис. 4.8).



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

Рис. 4.8: Поменяла владельца программы

Выполнила проверку правильности установки новых атрибутов и смены владельца файла simpleid2 (рис. 4.9).

```
[guest@localhost ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 26064 Mar 11 16:20 simpleid2
[guest@localhost ~]$
```

Рис. 4.9: ls -l

Запустила simpleid2 и id. Результаты похожи. Gid и uid одинаковые, однако команда id дает больше информации (рис. 4.10).

```
[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. 4.10: Сравнение результатов

Создала программу readfile.c (рис. 4.11).

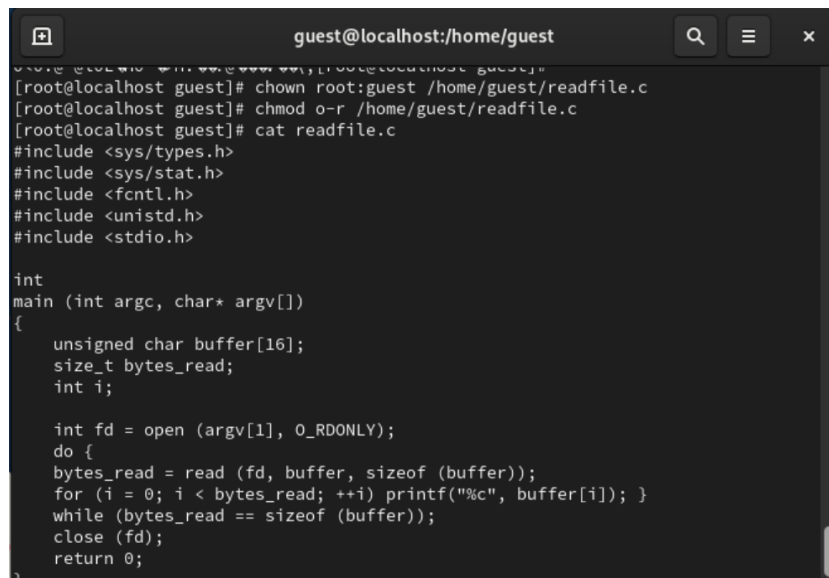
```
[guest@localhost ~]$ cat readfile.c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;

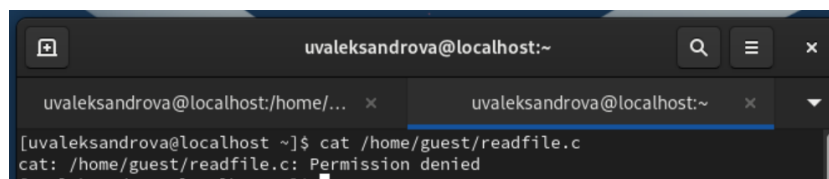
    int fd = open (argv[1], O_RDONLY);
    do {
        bytes_read = read (fd, buffer, sizeof (buffer));
        for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]); }
    while (bytes_read == sizeof (buffer));
    close (fd);
    return 0;
}
```

Рис. 4.11: Создание программы

Откомпилировала её (рис. 4.12).

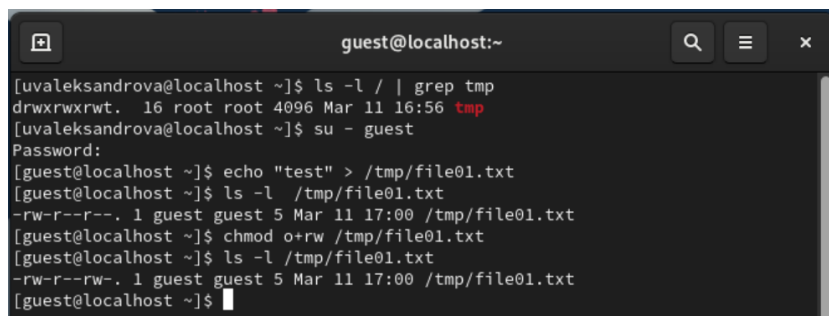


Сменила владельца у файла `readfile.c` и изменила права так, чтобы только суперпользователь (`root`) мог прочитать его, а `aleksandrova.uv` не мог. Проверила, может ли пользователь прочитать файл `readfile.c`. Не может (рис. 4.13).



4.2 Исследование Sticky-бита

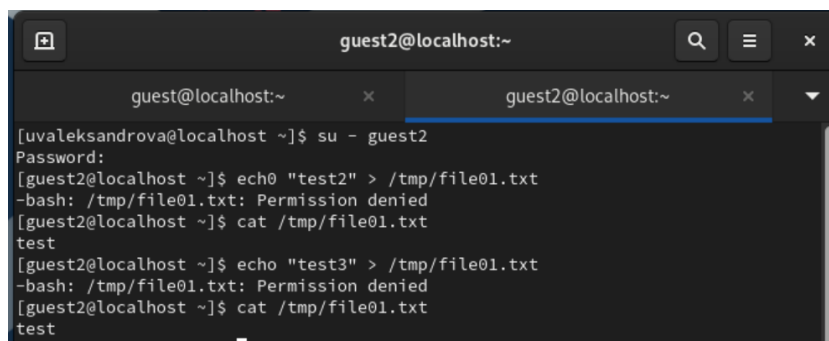
Выяснила, установлен ли атрибут Sticky на директории /tmp. Установлен. От имени пользователя guest создала файл file01.txt в директории /tmp со словом test через **echo "test" > /tmp/file01.txt**. Затем просмотрите атрибуты у только что созданного файла и разрешила чтение и запись для категории пользователей «все остальные» при помощи утилиты **chmod o+rw /tmp/file01.txt** (рис. 4.15).



```
guest@localhost:~  
[uvaleksandrova@localhost ~]$ ls -l / | grep tmp  
drwxrwxrwt. 16 root root 4096 Mar 11 16:56 tmp  
[uvaleksandrova@localhost ~]$ su - guest  
Password:  
[guest@localhost ~]$ echo "test" > /tmp/file01.txt  
[guest@localhost ~]$ ls -l /tmp/file01.txt  
-rw-r--r--. 1 guest guest 5 Mar 11 17:00 /tmp/file01.txt  
[guest@localhost ~]$ chmod o+rw /tmp/file01.txt  
[guest@localhost ~]$ ls -l /tmp/file01.txt  
-rw-r--rw-. 1 guest guest 5 Mar 11 17:00 /tmp/file01.txt  
[guest@localhost ~]$
```

Рис. 4.15: Выполнение задач

От пользователя guest2 (не являющегося владельцем) попробовала прочитать файл /tmp/file01.txt. Попробовала дозаписать в файл /tmp/file01.txt слово test2 командой **echo "test2" > /tmp/file01.txt**. Мне отказано в доступе. То же самое попробовала сделать с test3, но мне снова отказано в доступе. Файл не записался (рис. 4.16).

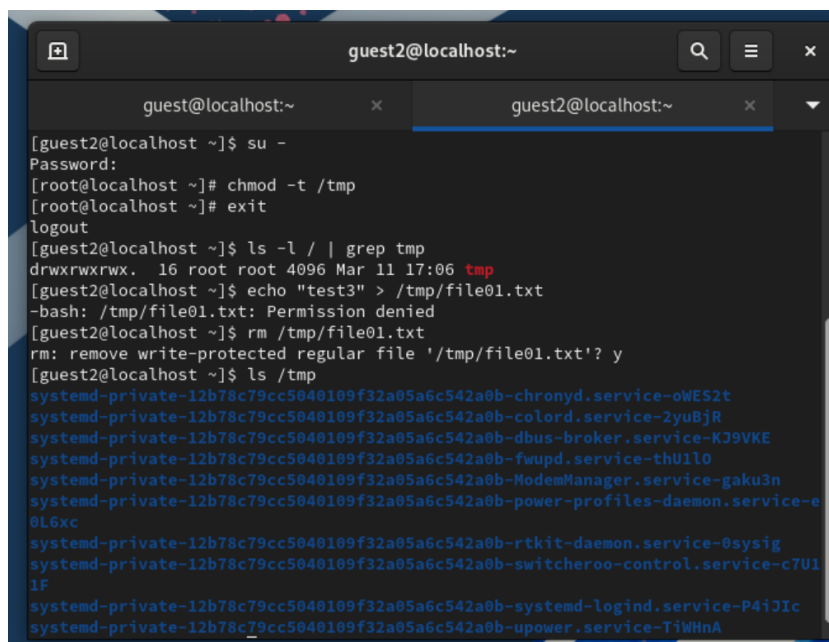


```
guest2@localhost:~  
[uvaleksandrova@localhost ~]$ su - guest2  
Password:  
[guest2@localhost ~]$ echo "test2" > /tmp/file01.txt  
-bash: /tmp/file01.txt: Permission denied  
[guest2@localhost ~]$ cat /tmp/file01.txt  
test  
[guest2@localhost ~]$ echo "test3" > /tmp/file01.txt  
-bash: /tmp/file01.txt: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

Рис. 4.16: Работа с файлами

От пользователя guest2 попробовала удалить файл /tmp/file01.txt. Мне отказали в доступе. Потом я повысила свои права до суперпользователя командой

su - и сняла атрибут **t** (Sticky-бит) с директории **/tmp**. От пользователя **guest2** проверила, что атрибута **t** у директории **/tmp** нет. Повторила предыдущие шаги. Файл удалился (рис. 4.17).



```
guest2@localhost:~  
[guest2@localhost ~]$ su -  
Password:  
[root@localhost ~]# chmod -t /tmp  
[root@localhost ~]# exit  
logout  
[guest2@localhost ~]$ ls -l / | grep tmp  
drwxrwxrwx. 16 root root 4096 Mar 11 17:06 tmp  
[guest2@localhost ~]$ echo "test3" > /tmp/file01.txt  
-bash: /tmp/file01.txt: Permission denied  
[guest2@localhost ~]$ rm /tmp/file01.txt  
rm: remove write-protected regular file '/tmp/file01.txt'? y  
[guest2@localhost ~]$ ls /tmp  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-chronyd.service-oWES2t  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-colord.service-2yuBJR  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-dbus-broker.service-KJ9VKE  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-fwupd.service-thU1l0  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-ModemManager.service-gaku3n  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-power-profiles-daemon.service-e0L6xc  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-rtkit-daemon.service-0sysig  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-switcheroo-control.service-c7U11F  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-systemd-logind.service-P4i3Ic  
systemd-private-12b78c79cc5040109f32a05a6c542a0b-upower.service-TiWHnA
```

Рис. 4.17: Работа с атрибутом **t**

Вернула атрибут **t** (рис. 4.18).



```
[root@localhost ~]# chmod +t /tmp  
[root@localhost ~]# exit
```

Рис. 4.18: Возвращение Sticky

5 Выводы

Я изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получила практических навыков работы в консоли с дополнительными атрибутами.