

# Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

---

Буллет Т. А.

15 февраля 2025

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Буллер Татьяна Александровна
- студент направления Бизнес-информатика
- Российский университет дружбы народов

## Вводная часть

---

- Операционная система linux, дистрибутив Rocky
- Среда виртуализации VirtualBox

- 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.

- Процессор **pandoc** для входного формата Markdown
- Среда виртуализации VirtualBox

## Выполнение лабораторной работы

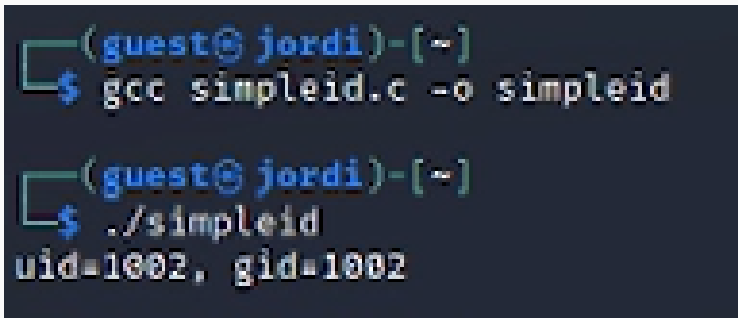
---



Создадим программу simpleid.c. Эта программа с помощью функций `geteuid` и `getegid` получает `uid` и `gid` пользователя соответственно, после чего выводит их на экран.

```
GNU nano 6.3 simplei
#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;
}
```

Скомпилировав программу, получим вывод, в значениях совпадающий с выводом команды `id`.

A terminal window with a dark background and light blue text. The prompt is `(guest@ jordi)-[~]`. The first command is `$ gcc simpleid.c -o simpleid`. The second command is `$ ./simpleid`, which outputs `uid=1002, gid=1002`.

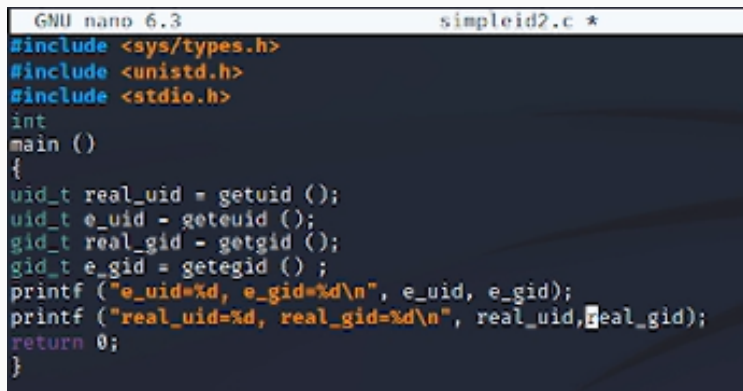
```
(guest@ jordi)-[~]  
$ gcc simpleid.c -o simpleid  
  
(guest@ jordi)-[~]  
$ ./simpleid  
uid=1002, gid=1002
```

Рис. 2: Скомпилированная программа

A terminal window with a dark background and light blue text. The prompt is `(guest@ jordi)-[~]`. The command is `$ ./simpleid`.

```
(guest@ jordi)-[~]  
$ ./simpleid
```

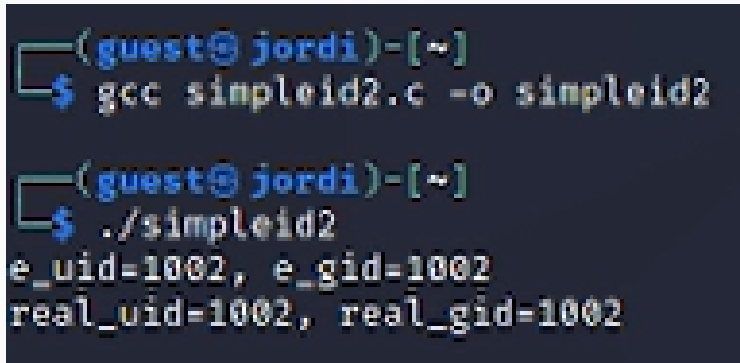
После этого усложним программу, как показано на скриншоте:



```
GNU nano 6.3                                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: simpleid2

Теперь вывод дополнен и все еще совпадает с выводом `id`.

A terminal window with a dark background and light-colored text. The prompt is (guest@jordi)-[~]. The first command is \$ gcc simpleid2.c -o simpleid2. The second prompt is (guest@jordi)-[~]. The second command is \$ ./simpleid2. The output of the program is e\_uid=1002, e\_gid=1002 followed by real\_uid=1002, real\_gid=1002 on the next line.

```
(guest@jordi)-[~]  
$ gcc simpleid2.c -o simpleid2  
  
(guest@jordi)-[~]  
$ ./simpleid2  
e_uid=1002, e_gid=1002  
real_uid=1002, real_gid=1002
```

Рис. 5: Дополненный вывод

Следующим шагом от имени суперпользователя назначим владельцам файла программы суперпользователя и добавим ей SUID-бит.

```
(tabuller@jordi)-[/home/guest]
$ sudo chown tabuller:guest /home/guest/simpleid2
[sudo] password for tabuller:

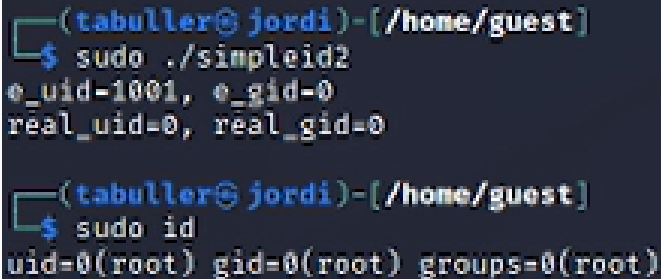
(tabuller@jordi)-[/home/guest]
$ sudo chmod u+s /home/guest/simpleid2
```

Рис. 6: Переназначение владельца и добавление SUID-бита

```
(guest@jordi)-[~]
$ ls -l simpleid2
-rwsr-xr-x 1 tabuller guest 16168 Feb 14 08:05 simpleid2
```

Рис. 7: Новые права программы

Теперь при запуске этой программы видим, что она выводит `e_uid`: идентификатор пользователя, от имени которого она была запущена; `real_uid` - идентификатор пользователя, от имени которого она выполняется.



```
(tabuller@jordi)-[/hone/guest]
$ sudo ./simpleid2
e_uid=1001, e_gid=0
real_uid=0, real_gid=0

(tabuller@jordi)-[/hone/guest]
$ sudo id
uid=0(root) gid=0(root) groups=0(root)
```

Рис. 8: Вывод `simpleid2` с SUID-битом

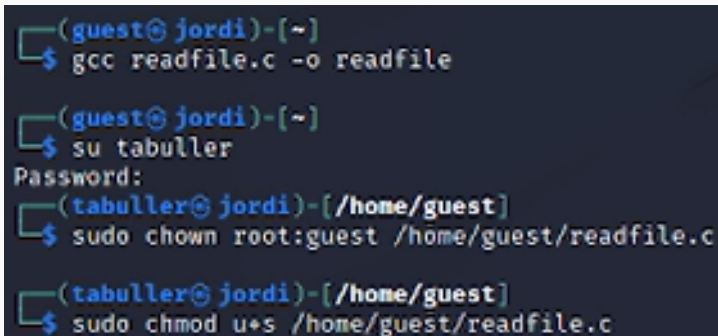
## Исследование SUID-бита

Создадим еще одну программу: аналог cat readfile, которая будет получать содержимое файла, название которого передано ей в аргументе, и выводить его на экран.

```
GNU nano 6.3                                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));
```

## Исследование SUID-бита

Скомпилируем ее, передадим файл кода во владение суперпользователю и добавим SUID-бит, после чего заберем у всех остальных пользователей все права на него. Видим, что теперь пользователь guest не может прочитать содержимое.



```
(guest@ jordi)-[~]  
$ gcc readfile.c -o readfile  
  
(guest@ jordi)-[~]  
$ su tabuller  
Password:  
(tabuller@ jordi)-[/home/guest]  
$ sudo chown root:guest /home/guest/readfile.c  
  
(tabuller@ jordi)-[/home/guest]  
$ sudo chmod u+s /home/guest/readfile.c
```

Рис. 10: Изменение прав файла кода



## Исследование SUID-бита

Следующим шагом изменим права на программу, которая была скомпилирована по коду readfile.c. Добавим тот же SUID-бит и передадим во владение суперпользователю. Вилим, что теперь название программы подсвечено красным.

```
(tabuller@jordi)-[/home/guest]
$ sudo chown root:guest readfile
[sudo] password for tabuller:

(tabuller@jordi)-[/home/guest]
$ ls -l readfile
ls: cannot access 'readfile': Permission denied

(tabuller@jordi)-[/home/guest]
$ exit
exit

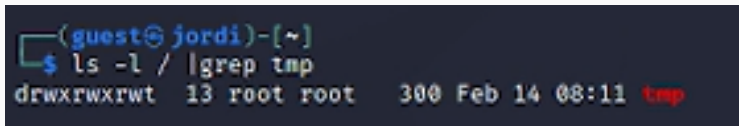
(guest@jordi)-[~]
$ ls -l readfile
-rwxr-xr-x 1 root guest 16104 Feb 14 08:11 readfile
```

## Исследование SUID-бита

Теперь при попытке прочитать `readfile.c` с помощью скомпилированной им программы мы получаем содержимое файла. Это происходит потому, что программа выполняется от имени суперпользователя и наделена всеми теми же правами, что и `root`. Она в том числе может прочитать и системный файл `/etc/shadow`, в котором по умолчанию хранится список пользователей и их хэшированные пароли.

```
(guest@jordi)-[~]  
$ ./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;  
}  
  
(guest@jordi)-[~]  
$ ./readfile /etc/shadow  
root:!:20120:0:00000:7:...
```

Просмотрев в корневой директории системы права поддиректорий, видим, что на директории tmp установлен бит t. Полезно отметить также, что писать и читать эту директорию может кто угодно.



```
(guest@jordi)-[~]  
$ ls -l / | grep tmp  
drwxrwxrwt 13 root root 300 Feb 14 08:11 tmp
```

Рис. 15: Sticky-бит на директории tmp

Далее создадим в этой директории файл file01. Изначально права на него позволяют пользователям кроме владельца только читать его, владельцу - записывать и читать. Добавим для остальных пользователей права на запись.

```
(guest@jordi)-[~]  
$ echo "test" > /tmp/file01.txt  
  
(guest@jordi)-[~]  
$ ls -l /tmp/file01.txt  
-rw-r--r-- 1 guest guest 5 Feb 14 08:18 /tmp/file01.txt  
  
(guest@jordi)-[~]  
$ chmod o+rw /tmp/file01.txt  
  
(guest@jordi)-[~]  
$ ls -l /tmp/file01.txt  
-rw-r--rw- 1 guest guest 5 Feb 14 08:18 /tmp/file01.txt
```

Рис. 16: Созданный в директории tmp файл

## Исследование Sticky-бита

Переключившись на пользователя guest2, однако, мы все еще не можем сделать с этим файлом ничего, кроме прочтения, потому что пользователь guest2 входит в группу пользователя guest, а группе не были добавлены права на запись. Файл, кроме прочего, “защищен” Sticky-битом, установленным на директории.

```
(guest2@jordi)-[/home/guest]
$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied

(guest2@jordi)-[/home/guest]
$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied

(guest2@jordi)-[/home/guest]
$ cat /tmp/file01.txt
test

(guest2@jordi)-[/home/guest]
$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory

(guest2@jordi)-[/home/guest]
```

## Исследование Sticky-бита

От имени суперпользователя снимем Sticky-бит и снова проверим права на директорию tmp: символ t пропал.



```
(root@jordi)-[/home/guest]
$ chmod -t /tmp

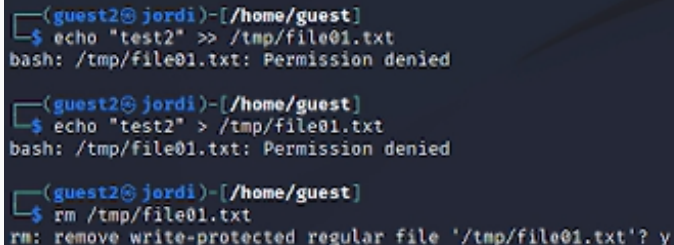
(root@jordi)-[/home/guest]
$ exit

(tabuller@jordi)-[/home/guest]
$ exit
exit

(guest2@jordi)-[/home/guest]
$ ls -l / | grep tmp
drwxrwxrwx  13 root root   320 Feb 14 08:18 tmp
```

Рис. 18: Снятие Sticky-бита

Теперь уже, все еще не являясь владельцем файла, мы можем его переименовывать и удалять.



```
(guest2@ jordi)-[/home/guest]
$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied

(guest2@ jordi)-[/home/guest]
$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied

(guest2@ jordi)-[/home/guest]
$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
```

Рис. 19: Работа с файлом от лица стороннего пользователя

## Выводы

---



Изучены механизмы изменения идентификаторов, применения SetUID- и Sticky-битов.  
Получены практические навыки работы в консоли с дополнительными атрибутами.  
Рассмотрена работа механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.