

Отчёт по лабораторной работе №5

**Дискреционное разграничение прав в Linux. Исследование влияния
дополнительных атрибутов**

Герра Гарсия Максимиано Антонио

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
2.1	Подготовка	5
2.2	Изучение механики SetUID	6
2.3	Исследование Sticky-бита	10
3	Выводы	13
	Список литературы	14

List of Figures

2.1	подготовка к работе	5
2.2	программа simpleid	6
2.3	результат программы simpleid	6
2.4	программа simpleid2	7
2.5	результат программы simpleid2	8
2.6	программа readfile	9
2.7	результат программы readfile	10
2.8	исследование Sticky-бита	12

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.

2.2 Изучение механики SetUID

1. Вошли в систему от имени пользователя guest.
2. Написали программу simpleid.c.

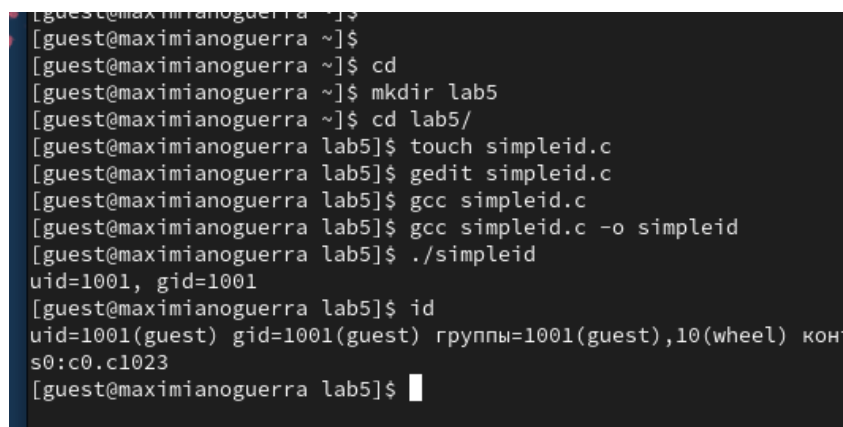


```
simpleid.c
~/lab5

1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int main()
5 {
6     uid_t uid = geteuid();
7     gid_t gid = getegid();
8     printf("uid=%d, gid=%d\n", uid, gid);
9     return 0;
10 }
```

Figure 2.2: программа simpleid

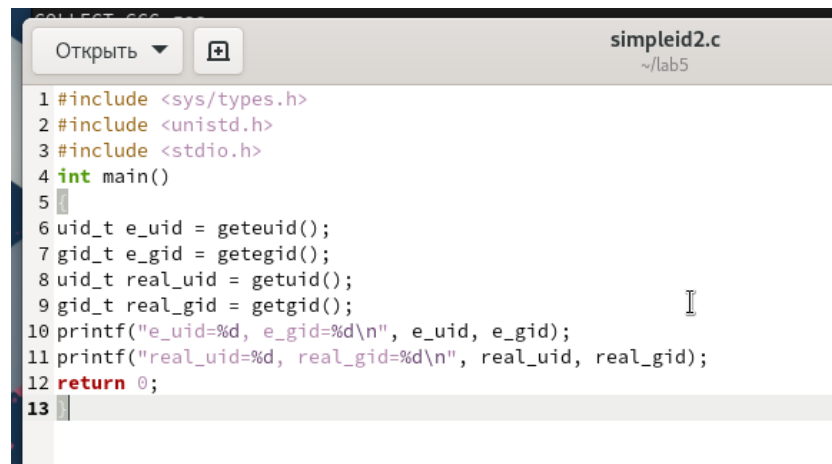
3. Скомпилировали программу и убедились, что файл программы создан: `gcc simpleid.c -o simpleid`
4. Выполнили программу simpleid командой `./simpleid`
5. Выполнили системную программу id с помощью команды `id`. uid и gid совпадает в обеих программах



```
[guest@maximianoquerra ~]$
[guest@maximianoquerra ~]$
[guest@maximianoquerra ~]$ cd
[guest@maximianoquerra ~]$ mkdir lab5
[guest@maximianoquerra ~]$ cd lab5/
[guest@maximianoquerra lab5]$ touch simpleid.c
[guest@maximianoquerra lab5]$ gedit simpleid.c
[guest@maximianoquerra lab5]$ gcc simpleid.c
[guest@maximianoquerra lab5]$ gcc simpleid.c -o simpleid
[guest@maximianoquerra lab5]$ ./simpleid
uid=1001, gid=1001
[guest@maximianoquerra lab5]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest),10(wheel) конс0:c0.c1023
[guest@maximianoquerra lab5]$
```

Figure 2.3: результат программы simpleid

6. Усложнили программу, добавив вывод действительных идентификаторов.



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int main()
5 {
6     uid_t e_uid = geteuid();
7     gid_t e_gid = getegid();
8     uid_t real_uid = getuid();
9     gid_t real_gid = getgid();
10    printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
11    printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
12    return 0;
13 }
```

Figure 2.4: программа simpleid2

7. Скомпилировали и запустили simpleid2.c:

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

```
./simpleid2
```

8. От имени суперпользователя выполнили команды:

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

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

9. Использовали su для повышения прав до суперпользователя

10. Выполнили проверку правильности установки новых атрибутов и смены владельца файла simpleid2:

```
ls -l simpleid2
```

11. Запустили simpleid2 и id:

```
./simpleid2
```

```
id
```

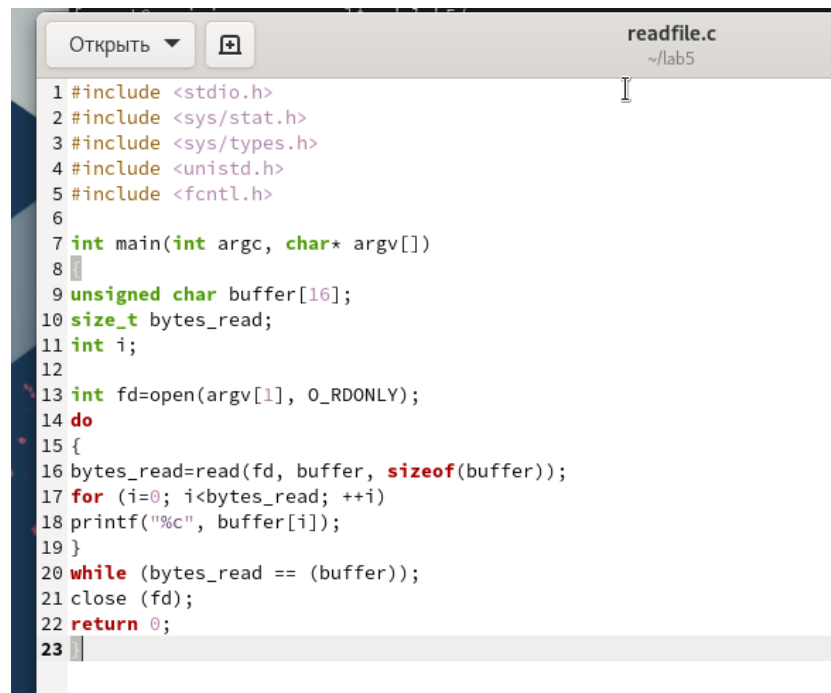
Результат выполнения программ теперь немного отличается

12. Проделали тоже самое относительно SetGID-бита.

```
[guest@maximianoguerra lab5]$
[guest@maximianoguerra lab5]$ touch simleid2.c
[guest@maximianoguerra lab5]$ mv simleid2.c simpleid2.c
[guest@maximianoguerra lab5]$ gedit simpleid2.c
[guest@maximianoguerra lab5]$ gcc simpleid2.c
[guest@maximianoguerra lab5]$ gcc simpleid2.c -o simpleid2
[guest@maximianoguerra lab5]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@maximianoguerra lab5]$ su
Пароль:
[root@maximianoguerra lab5]# chown root:guest simpleid2
[root@maximianoguerra lab5]# chmod u+s simpleid2
[root@maximianoguerra lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@maximianoguerra lab5]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:unconfined_s
[root@maximianoguerra lab5]# chmod g+s simpleid2
[root@maximianoguerra lab5]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@maximianoguerra lab5]#
exit
[guest@maximianoguerra lab5]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@maximianoguerra lab5]$
```

Figure 2.5: результат программы simpleid2

13. Написали программу readfile.c

The image shows a code editor window with a title bar that includes a dropdown menu labeled 'Открыть' (Open) and a button with a plus sign. The file name 'readfile.c' and its path '~ /lab5' are displayed in the top right corner. The code is written in C and includes several header files: <stdio.h>, <sys/stat.h>, <sys/types.h>, <unistd.h>, and <fcntl.h>. The main function takes two arguments, argc and argv. It declares a buffer of 16 unsigned characters, a size_t variable for bytes_read, and an integer i. It opens the file specified in argv[1] in read-only mode. A do-while loop reads the file content into the buffer and prints it character by character. The loop continues until the read operation returns the size of the buffer. Finally, it closes the file and returns 0.

```
1 #include <stdio.h>
2 #include <sys/stat.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <fcntl.h>
6
7 int main(int argc, char* argv[])
8 {
9     unsigned char buffer[16];
10    size_t bytes_read;
11    int i;
12
13    int fd=open(argv[1], O_RDONLY);
14    do
15    {
16        bytes_read=read(fd, buffer, sizeof(buffer));
17        for (i=0; i<bytes_read; ++i)
18            printf("%c", buffer[i]);
19    }
20    while (bytes_read == (buffer));
21    close (fd);
22    return 0;
23 }
```

Figure 2.6: программа readfile

14. Откомпилировали её.

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

15. Сменили владельца у файла readfile.c и изменили права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

```
chown root:guest /home/guest/readfile.c
```

```
chmod 700 /home/guest/readfile.c
```

16. Проверили, что пользователь guest не может прочитать файл readfile.c.

17. Сменили у программы readfile владельца и установили SetU'D-бит.

18. Проверили, может ли программа readfile прочитать файл readfile.c

19. Проверили, может ли программа readfile прочитать файл /etc/shadow

```
[guest@maximianoguerra lab5]$ touch readfile.c
[guest@maximianoguerra lab5]$ gedit readfile.c
[guest@maximianoguerra lab5]$ gcc readfile.c
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
20 | while (bytes_read == (buffer));
    |                   ^~
[guest@maximianoguerra lab5]$ gcc readfile.c -o readfile
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
20 | while (bytes_read == (buffer));
    |                   ^~
[guest@maximianoguerra lab5]$ su
Пароль:
[root@maximianoguerra lab5]# chown root:root readfile
[root@maximianoguerra lab5]# chmod -rwx readfile.c
[root@maximianoguerra lab5]# chmod u+s readfile
[root@maximianoguerra lab5]#
exit
[guest@maximianoguerra lab5]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@maximianoguerra lab5]$ ./readfile readfile.c
#include <stdio.h>
[guest@maximianoguerra lab5]$ ./readfile /etc/shadow
root:$6$0mJpklj[guest@maximianoguerra lab5]$
[guest@maximianoguerra lab5]$
```

Figure 2.7: результат программы readfile

2.3 Исследование Sticky-бита

1. Выяснили, установлен ли атрибут Sticky на директории /tmp:

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

2. От имени пользователя guest создали файл file01.txt в директории /tmp со словом test:

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

3. Просмотрели атрибуты у только что созданного файла и разрешили чтение и запись для категории пользователей «все остальные»:

```
ls -l /tmp/file01.txt
```

```
chmod o+rw /tmp/file01.txt
```

```
ls -l /tmp/file01.txt
```

Первоначально все группы имели право на чтение, а запись могли осуществлять все, кроме «остальных пользователей».

4. От пользователя (не являющегося владельцем) попробовали прочитать файл /file01.txt:

```
cat /file01.txt
```

5. От пользователя попробовали дозаписать в файл /file01.txt слово test3 командой:

```
echo "test2" >> /file01.txt
```

6. Проверили содержимое файла командой:

```
cat /file01.txt
```

В файле теперь записано:

```
Test
```

```
Test2
```

7. От пользователя попробовали записать в файл /tmp/file01.txt слово test4, стерев при этом всю имеющуюся в файле информацию командой. Для этого воспользовалась командой `echo "test3" > /tmp/file01.txt`

8. Проверили содержимое файла командой

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

9. От пользователя попробовали удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`, однако получила отказ.
10. От суперпользователя командой выполнили команду, снимающую атрибут `t` (Sticky-бит) с директории /tmp:

```
chmod -t /tmp
```

Покинули режим суперпользователя командой `exit`.

11. От пользователя проверили, что атрибута `t` у директории `/tmp` нет:

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

12. Повторили предыдущие шаги. Получилось удалить файл

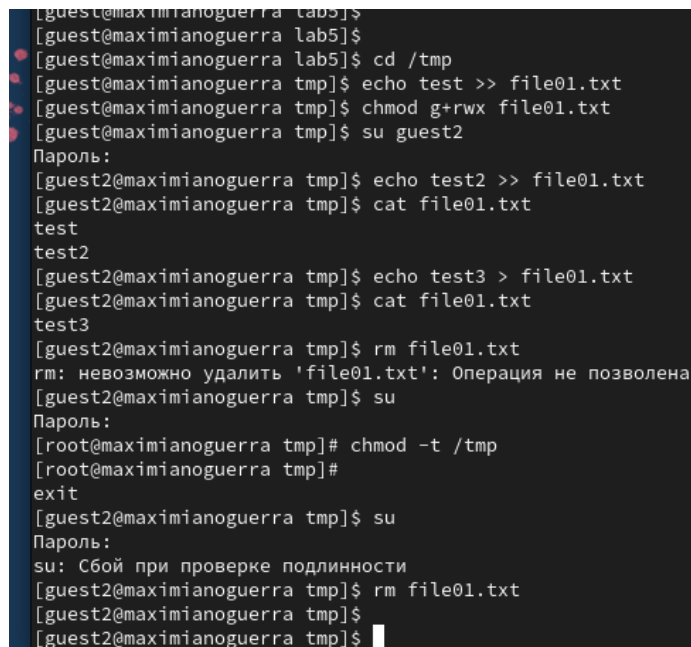
13. Удалось удалить файл от имени пользователя, не являющегося его владельцем.

14. Повысили свои права до суперпользователя и вернули атрибут `t` на директорию `/tmp` :

```
su
```

```
chmod +t /tmp
```

```
exit
```



```
[guest@maximiano guerra lab5]$  
[guest@maximiano guerra lab5]$  
[guest@maximiano guerra lab5]$ cd /tmp  
[guest@maximiano guerra tmp]$ echo test >> file01.txt  
[guest@maximiano guerra tmp]$ chmod g+rw file01.txt  
[guest@maximiano guerra tmp]$ su guest2  
Пароль:  
[guest2@maximiano guerra tmp]$ echo test2 >> file01.txt  
[guest2@maximiano guerra tmp]$ cat file01.txt  
test  
test2  
[guest2@maximiano guerra tmp]$ echo test3 > file01.txt  
[guest2@maximiano guerra tmp]$ cat file01.txt  
test3  
[guest2@maximiano guerra tmp]$ rm file01.txt  
rm: невозможно удалить 'file01.txt': Операция не позволена  
[guest2@maximiano guerra tmp]$ su  
Пароль:  
[root@maximiano guerra tmp]# chmod -t /tmp  
[root@maximiano guerra tmp]#  
exit  
[guest2@maximiano guerra tmp]$ su  
Пароль:  
su: Сбой при проверке подлинности  
[guest2@maximiano guerra tmp]$ rm file01.txt  
[guest2@maximiano guerra tmp]$  
[guest2@maximiano guerra tmp]$
```

Figure 2.8: исследование Sticky-бита

3 Выводы

Изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Также мы рассмотрели работу механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов.

Список литературы

1. КОМАНДА CHATTR В LINUX
2. chattr