

# **Отчет по лабораторной работе №5**

**Информационная безопасность**

Коломиец Мария Владимировна НПИбд-01-18

# Содержание

1	Цель работы	4
2	Теоретическое описание	5
3	Выполнение лабораторной работы	6
4	Выводы	14

## Список иллюстраций

3.1	Создание программы simpleid.c . . . . .	6
3.2	Компиляция, выполнение программы . . . . .	7
3.3	Создание программы simpleid2.c . . . . .	7
3.4	Компиляция, выполнение программы . . . . .	8
3.5	Выполнение . . . . .	8
3.6	Выполнение . . . . .	8
3.7	Установка атрибутов . . . . .	8
3.8	Проверка . . . . .	9
3.9	Создание программы readfile.c . . . . .	9
3.10	Создание программы readfile.c . . . . .	9
3.11	Изменение владельца и прав . . . . .	9
3.12	Проверка . . . . .	10
3.13	Изменение для программы readfile . . . . .	10
3.14	Проверка . . . . .	10
3.15	Проверка . . . . .	11
3.16	Выполнение . . . . .	11
3.17	Выполнение и проверка от пользователя guest2 . . . . .	12
3.18	Снятие атрибута “t” с директории /tmp . . . . .	12
3.19	Проверка . . . . .	12
3.20	Добавление атрибута “t” на директорию /tmp . . . . .	12
3.21	Проверка . . . . .	13

# 1 Цель работы

Изучить механизмы изменения идентификаторов, применение SetUID- и Sticky-битов. Получить практические навыки работы в консоли с дополнительными атрибутами. Рассмотреть работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

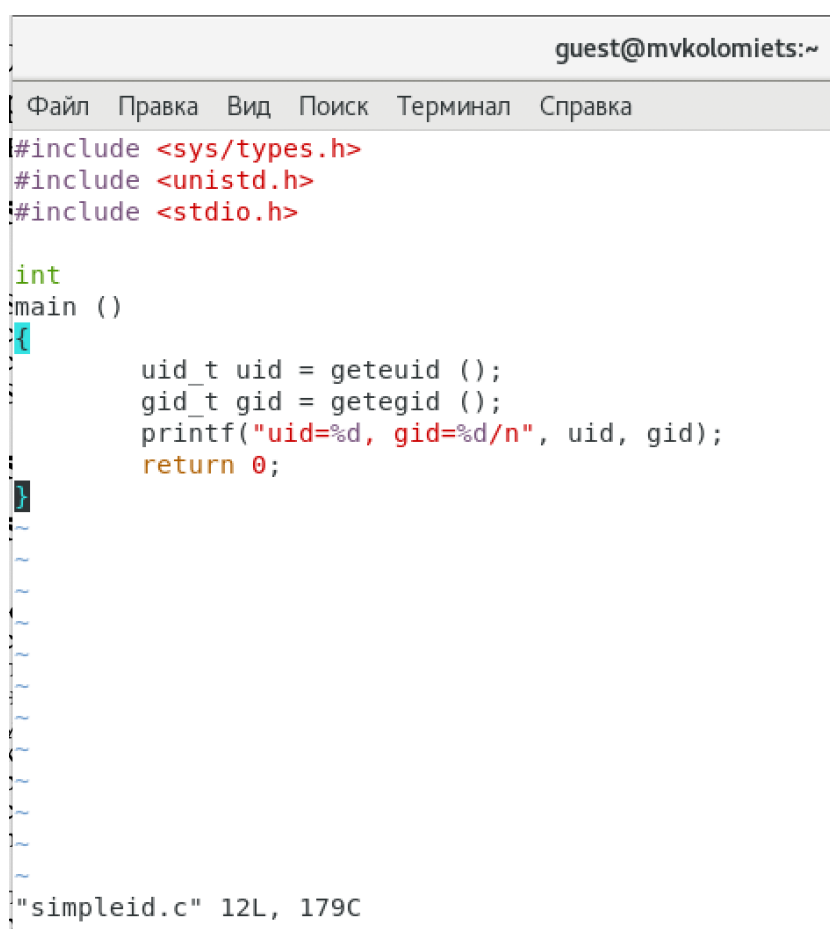
## 2 Теоретическое описание

В Linux, как и в любой многопользовательской системе, абсолютно естественным образом возникает задача разграничения доступа субъектов — пользователей к объектам — файлам дерева каталогов.

Setuid, Setgid и Sticky Bit - это специальные типы разрешений позволяют задавать расширенные права доступа на файлы или каталоги. Setuid – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволяет нам поднять привилегии пользователя в случае, если это необходимо. Принцип работы Setgid очень похож на setuid с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом. Последний специальный бит разрешения – это Sticky Bit . В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем.

### 3 Выполнение лабораторной работы

1. Вошла в систему от имени пользователя guest, создала программу simpleid.c. (рис. 3.1).



```
guest@mvkolomiets:~  
Файл Правка Вид Поиск Терминал Справка  
#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;  
}  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
"simpleid.c" 12L, 179C
```

Рис. 3.1: Создание программы simpleid.c

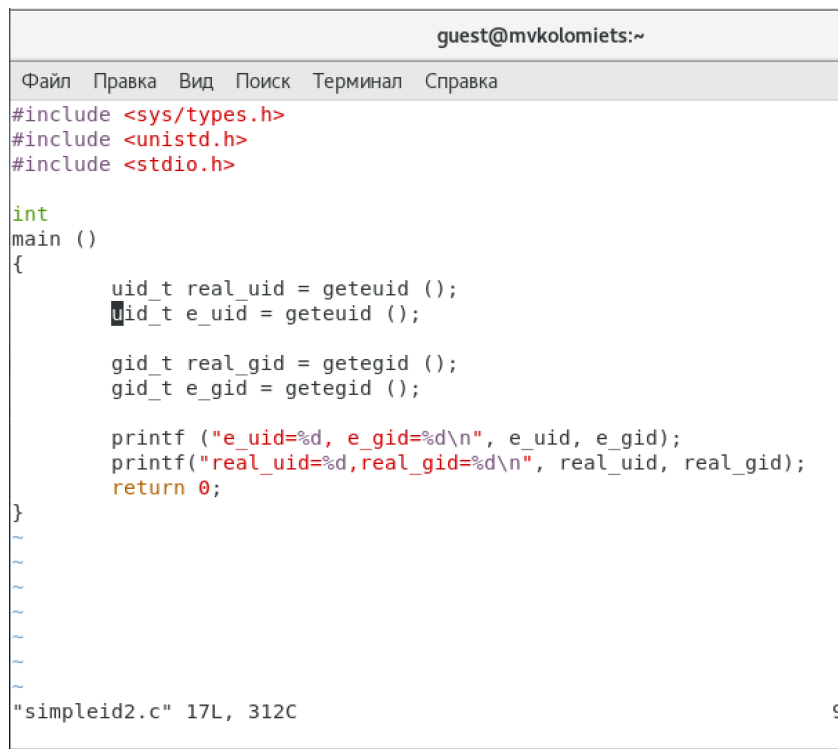
2. Скомпилировала программу и убедилась, что файл программы создан: gcc simpleid.c -o simpleid. Выполнила программу simpleid: ./simpleid. Выполнила

системную программу `id`. И сравнила полученный результат с данными предыдущего пункта задания. (Данные одинаковы)(рис. 3.2).

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

Рис. 3.2: Компиляция, выполнение программы

3. Усложнила программу, добавив вывод действительных идентификаторов. (рис. 3.3).



```
guest@mvkolomiets:~
Файл Правка Вид Поиск Терминал Справка
#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;
}

"simpleid2.c" 17L, 312C 9
```

Рис. 3.3: Создание программы `simpleid2.c`

4. Скомпилировала и запустила `simpleid2.c`: `gcc simpleid2.c -o simpleid2; ./simpleid2` (рис. 3.4).

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

Рис. 3.4: Компиляция, выполнение программы

- От имени суперпользователя выполнила команды: `chown root:guest /home/guest/simpleid2`; `chmod u+s /home/guest/simpleid2`. (рис. 3.5).

```
[mvkolomiets@mvkolomiets ~]$ su
Пароль:
[root@mvkolomiets mvkolomiets]# cd ~
[root@mvkolomiets ~]# chown root:guest /home/guest/simpleid2
[root@mvkolomiets ~]# chmod u+s /home/guest/simpleid2
```

Рис. 3.5: Выполнение

С помощью первой команды для файла `simpleid2.c` мы поменяли пользователя и группу на `root` и `guest` соответственно. С помощью второй установили разрешение для пользователей на выполнение с разрешением владельца.

- Выполнила проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`: `ls -l simpleid2`. Запустила `simpleid2` и `id`. (рис. 3.6).

```
[guest@mvkolomiets ~]$ ls -l simpleid2
-rwsrwsr-x. 1 root guest 8472 ноя  9 18:34 simpleid2
[guest@mvkolomiets ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=0,real_gid=1001
[guest@mvkolomiets ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 3.6: Выполнение

- Проделала тоже самое относительно `SetGID`-бита. (рис. 3.7), (рис. 3.8).

```
[root@mvkolomiets ~]# chmod g+s /home/guest/simpleid2
[root@mvkolomiets ~]#
```

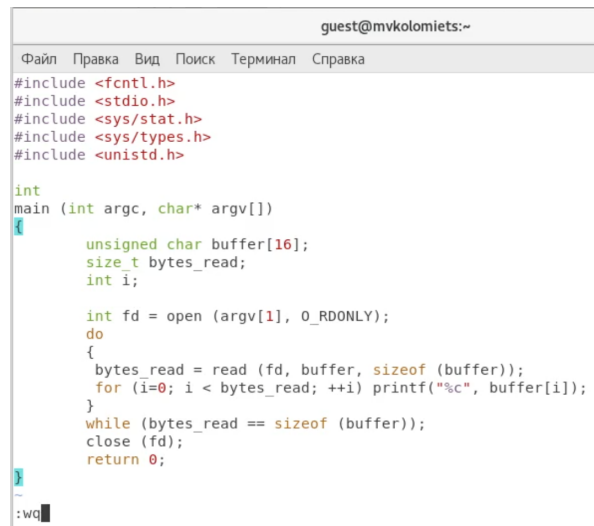
Рис. 3.7: Установка атрибутов



```
[guest@mvkolomiets ~]$ ls -l simpleid2
-rwsrwsr-x. 1 root guest 8472 ноя  9 18:34 simpleid2
[guest@mvkolomiets ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=1001
[guest@mvkolomiets ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 3.8: Проверка

## 8. Создала программу readfile.c: (рис. 3.9).



```
guest@mvkolomiets:~
Файл  Правка  Вид  Поиск  Терминал  Справка
#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;
}
```

Рис. 3.9: Создание программы readfile.c

## 9. Откомпилировала ее. (рис. 3.10).

```
[guest@mvkolomiets ~]$ gcc readfile.c -o readfile
```

Рис. 3.10: Создание программы readfile.c

## 10. Сменила владельца у файла readfile.c и изменила права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог. (рис. 3.11).

```
[root@mvkolomiets ~]# chown root /home/guest/readfile.c
[root@mvkolomiets ~]# chmod u+x /home/guest/readfile.c
[root@mvkolomiets ~]# chmod g-rw /home/guest/readfile.c
[root@mvkolomiets ~]# chmod o-r /home/guest/readfile.c
```

Рис. 3.11: Изменение владельца и прав

11. Проверила, что пользователь guest не может прочитать файл readfile.c. (рис. 3.12).

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

Рис. 3.12: Проверка

12. Сменила у программы readfile владельца и установила SetU'D-бит. (рис. 3.13).

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

Рис. 3.13: Изменение для программы readfile

13. Проверила, может ли программа readfile прочитать файл readfile.c (может), проверила, может ли программа readfile прочитать файл /etc/shadow (может). (рис. 3.14). (рис. 3.15).

```
[guest@mvkolomiets ~]$ ./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;
}
```

Рис. 3.14: Проверка

```
[guest@mvkolomiets ~]$ ./readfile /etc/shadow
root:$6$30o10cBwK3TmoZD$3xMwd13eyyoc/oZekM06E6f4C5xn/fts6UNDzYCWnyVZhQrJZZNz6CabEes
mTuodd1bCIaW3HFXPL.hwsN6Up1::0:99999:7:::
bin:*:18353:0:99999:7:::
daemon:*:18353:0:99999:7:::
adm:*:18353:0:99999:7:::
lp:*:18353:0:99999:7:::
sync:*:18353:0:99999:7:::
shutdown:*:18353:0:99999:7:::
halt:*:18353:0:99999:7:::
mail:*:18353:0:99999:7:::
operator:*:18353:0:99999:7:::
games:*:18353:0:99999:7:::
ftp:*:18353:0:99999:7:::
nobody:*:18353:0:99999:7:::
systemd-network:!:18884:!:
dbus:!:18884:!:
polkitd:!:18884:!:
libstoragemgmt:!:18884:!:
colord:!:18884:!:
rpc:!:18884:0:99999:7:::
saned:!:18884:!:
saslauth:!:18884:!:
abrt:!:18884:!:
setroubleshoot:!:18884:!:
rtkit:!:18884:!:
```

Рис. 3.15: Проверка

14. Выяснила, установлен ли атрибут Sticky на директории /tmp, для чего выполнила команду: `ls -l / | grep tmp`. От имени пользователя guest создала файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt`. Просмотрела атрибуты у только что созданного файла и разрешила чтение и запись для категории пользователей «все остальные»: `ls -l /tmp/file01.txt`, `chmod o+rw /tmp/file01.txt`, `ls -l /tmp/file01.txt`. (рис. 3.16).

```
[guest@mvkolomiets ~]$ ls -l / | grep tmp
drwxrwxrwt. 15 root root 4096 ноя  9 18:46 tmp
[guest@mvkolomiets ~]$ echo "test" > /tmp/file01.txt
[guest@mvkolomiets ~]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 ноя  9 19:05 /tmp/file01.txt
[guest@mvkolomiets ~]$ chmod o+rw /tmp/file01.txt
[guest@mvkolomiets ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 ноя  9 19:05 /tmp/file01.txt
```

Рис. 3.16: Выполнение

15. От пользователя guest2 (не являющегося владельцем) попробовала прочесть файл /tmp/file01.txt: `cat /tmp/file01.txt`, попробовала дозаписать в файл /tmp/file01.txt слово test2 командой: `echo "test2" > /tmp/file01.txt`. Проверила содержимое файла командой: `cat /tmp/file01.txt`. Также попробовала записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой: `echo "test3" > /tmp/file01.txt`. От пользователя guest2 попробовала удалить файл /tmp/file01.txt командой: `rm /tmp/file01.txt`. (Все действия, кроме удаления файла, выполнить удалось). (рис. 3.17).

```
[guest2@mvkolomiets ~]$ cat /tmp/file01.txt
test
[guest2@mvkolomiets ~]$ echo "test2" >> /tmp/file01.txt
[guest2@mvkolomiets ~]$ cat /tmp/file01.txt
test
test2
[guest2@mvkolomiets ~]$ echo "test3" > /tmp/file01.txt
[guest2@mvkolomiets ~]$ cat /tmp/file01.txt
test3
[guest2@mvkolomiets ~]$ rm /tmp/file01.txt
rm: невозможно удалить «/tmp/file01.txt»: Операция не позволена
```

Рис. 3.17: Выполнение и проверка от пользователя guest2

16. От суперпользователя выполнила команду, снимающую атрибут t (Sticky-бит) с директории /tmp: `chmod -t /tmp`. (рис. 3.18).

```
[root@mvkolomiets ~]# chmod -t /tmp
```

Рис. 3.18: Снятие атрибута “t” с директории /tmp

17. От пользователя guest2 проверила, что атрибута t у директории /tmp нет: `ls -l / | grep tmp`. Повторила предыдущие шаги. Нам удалось удалить файл от имени пользователя, не являющегося его владельцем, также получилось выполнить дозапись в файл и замену текста в файле. (рис. 3.19).

```
[guest2@mvkolomiets ~]$ ls -l / | grep tmp
drwxrwxrwx. 15 root root 4096 ноя  9 19:10 tmp
[guest2@mvkolomiets ~]$ cat /tmp/file01.txt
test3
[guest2@mvkolomiets ~]$ echo "test2" >> /tmp/file01.txt
[guest2@mvkolomiets ~]$ cat /tmp/file01.txt
test3
test2
[guest2@mvkolomiets ~]$ echo "test" > /tmp/file01.txt
[guest2@mvkolomiets ~]$ cat /tmp/file01.txt
test
[guest2@mvkolomiets ~]$ rm /tmp/file01.txt
[guest2@mvkolomiets ~]$ ls /tmp
ssh-6yofhz5Xszkd
systemd-private-476453378ee4461fa47c555c03b0dff7-bolt.service-kl70zQ
systemd-private-476453378ee4461fa47c555c03b0dff7-colord.service-UYC01l
systemd-private-476453378ee4461fa47c555c03b0dff7-cups.service-oUWSze
systemd-private-476453378ee4461fa47c555c03b0dff7-fwupd.service-kukI29
systemd-private-476453378ee4461fa47c555c03b0dff7-rtkit-daemon.service-ETeCa6
tracker-extract-files.1000
yum_save_tx.2021-11-09.18-01.912_ph.yumtx
```

Рис. 3.19: Проверка

18. От суперпользователя вернула атрибут t на директорию /tmp: `chmod +t /tmp`. (рис. 3.20), (рис. 3.21).

```
[root@mvkolomiets ~]# chmod +t /tmp
[root@mvkolomiets ~]# █
```

Рис. 3.20: Добавление атрибута “t” на директорию /tmp

```
[guest2@mvkolomiets ~]$ ls -l / | grep tmp  
drwxrwxrwt. 15 root root 4096 ноя  9 19:13 tmp  
[guest2@mvkolomiets ~]$ █
```

Рис. 3.21: Проверка

## 4 Выводы

На основе проделанной работы изучила механизмы изменения идентификаторов, применение 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.