

Лабораторная работа №5

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

Латыпова Диана. НФИбд-02-21

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	9
4	Выполнение лабораторной работы	10
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Установка gсс	10
4.2	Задание 1.1-5	11
4.3	Задание 1.6-7	12
4.4	Задание 1.8-12	13
4.5	Задание 1.13-14	14
4.6	Задание 1.13-14	14
4.7	Задание 1.16-19 (1)	15
4.8	Задание 1.16-19 (2)	16
4.9	Задание 2.1-4	17
4.10	Задание 2.5-14	18
4.11	Задание 2.15	19

Список таблиц

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 Задание

5.3.1. Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте программу simpleid.c
3. Скомпилируйте программу и убедитесь, что файл программы создан:

```
gcc simpleid.c -o simpleid
```

4. Выполните программу simpleid: ./simpleid
5. Выполните системную программу id: id и сравните полученный вами результат с данными предыдущего пункта задания.
6. Усложните программу, добавив вывод действительных идентификаторов. Получившуюся программу назовите simpleid2.c.
7. Скомпилируйте и запустите simpleid2.c:

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

8. От имени суперпользователя выполните команды

```
chown root:guest /home/guest/simpleid2
chmod u+s /home/guest/simpleid2
```

9. Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды.

10. Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2: `ls -l simpleid2`

11. Запустите simpleid2 и id:

```
./simpleid2
```

```
id
```

Сравните результаты. 12. Прodelайте тоже самое относительно SetGID-бита.

13. Создайте программу readfile.c. 14. Откомпилируйте её.

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

15. 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.

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

17. Смените у программы readfile владельца и установите SetU'D-бит.

18. Проверьте, может ли программа readfile прочитать файл readfile.c?

19. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отразите полученный результат и ваши объяснения в отчёте.

5.3.2. Исследование Sticky-бита

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду `ls -l / | grep tmp`

2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt`

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

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

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

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

4. От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt: `cat /tmp/file01.txt`
5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" > /tmp/file01.txt` Удалось ли вам выполнить операцию?
6. Проверьте содержимое файла командой `cat /tmp/file01.txt`
7. От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой `echo "test3" > /tmp/file01.txt` Удалось ли вам выполнить операцию?
8. Проверьте содержимое файла командой `cat /tmp/file01.txt`
9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt` Удалось ли вам удалить файл?
10. Повысьте свои права до суперпользователя следующей командой `su -` и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp:

```
chmod -t /tmp
```

11. Покиньте режим суперпользователя командой `exit`
12. От пользователя guest2 проверьте, что атрибута t у директории /tmp нет: `ls -l / | grep tmp`
13. Повторите предыдущие шаги. Какие наблюдаются изменения?
14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Ваши наблюдения занесите в отчёт.
15. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp:

```
su -
```

```
chmod +t /tmp
```

```
exit
```


3 Теоретическое введение

- **UID (User ID)** — идентификатор пользователя, который определяет, от имени какого пользователя выполняется программа[1].
- **GID (Group ID)** — идентификатор группы, определяющий группу, к которой принадлежит пользователь [1].

Программы могут выполняться с различными идентификаторами: действительный (реальный) и эффективный UID/GID. *Реальный UID* — это пользователь, который запустил процесс, а *эффективный UID* — это пользователь, от имени которого процесс выполняет операции.

- **SetUID** — специальный бит, который при установке на файл исполняемой программы позволяет запускать её с правами владельца файла, а не того пользователя, который её запускает [2].
- **SetGID** работает аналогично, но изменяет эффективный GID на GID владельца файла [2].

Sticky-bit — атрибут директории, который запрещает пользователям удалять файлы, если они не являются их владельцами, даже если у них есть права на запись в директорию[2].

4 Выполнение лабораторной работы

Для начала установили компилятор gcc (рис. 4.1):

```
[guest@user ~]$ su
Password:
[root@user guest]# yum install gcc
Extra Packages for Enterprise Linux 9 - x86_64 56 kB/s | 35 kB 00:00
Extra Packages for Enterprise Linux 9 - x86_64 4.9 MB/s | 23 MB 00:04
packages for the GitHub CLI 18 kB/s | 3.0 kB 00:00
packages for the GitHub CLI 10 kB/s | 2.7 kB 00:00
Rocky Linux 9 - BaseOS 13 kB/s | 4.1 kB 00:00
Rocky Linux 9 - BaseOS 2.1 MB/s | 2.3 MB 00:01
Rocky Linux 9 - AppStream 12 kB/s | 4.5 kB 00:00
Rocky Linux 9 - AppStream 6.3 MB/s | 8.0 MB 00:01
Rocky Linux 9 - Extras 9.3 kB/s | 2.9 kB 00:00
Visual Studio Code 11 kB/s | 1.5 kB 00:00
Visual Studio Code 2.6 MB/s | 5.1 MB 00:01
Package gcc-11.4.1-3.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@user guest]# setenforce 0
[root@user guest]# setenforce
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]
```

Рис. 4.1: Установка gcc

Вошла в систему от имени пользователя guest, создала программу simpleid.c:

```
#include <sys/types.h>    // Для использования типов данных UID и GID
#include <unistd.h>        // Для функций getuid(), geteuid(), getgid(), getegid()
#include <stdio.h>         // Для функции printf()

int main () {
    // Получение эффективных идентификаторов пользователя и группы
    uid_t uid = geteuid(); // Получение эффективного UID
    gid_t gid = getegid(); // Получение эффективного GID
```

```

// Вывод эффективных идентификаторов
printf("uid=%d, gid=%d\n", uid, gid);

return 0;
}

```

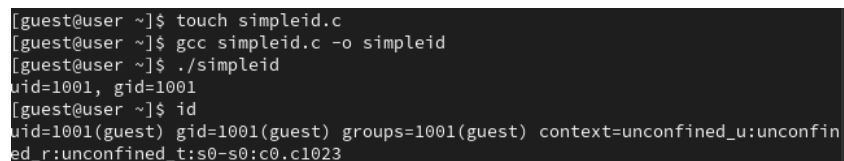
Скомпилировала программу, выполнила программу simpleid и системную программу id:

```

gcc simpleid.c -o simpleid
./simpleid
id

```

Видим, что UID и GID совпадают(рис. 4.2):



```

[guest@user ~]$ touch simpleid.c
[guest@user ~]$ gcc simpleid.c -o simpleid
[guest@user ~]$ ./simpleid
uid=1001, gid=1001
[guest@user ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

```

Рис. 4.2: Задание 1.1-5

Далее усложнила программу, добавив вывод действительных идентификаторов:

```

#include <sys/types.h>    // Для использования типов данных UID и GID
#include <unistd.h>        // Для функций getuid(), geteuid(), getgid(), getegid()
#include <stdio.h>         // Для функции printf()

int main () {
    // Получение реальных и эффективных идентификаторов пользователя и группы
    uid_t real_uid = getuid();    // Получение реального UID
    uid_t e_uid = geteuid();      // Получение эффективного UID
    gid_t real_gid = getgid();    // Получение реального GID

```

```

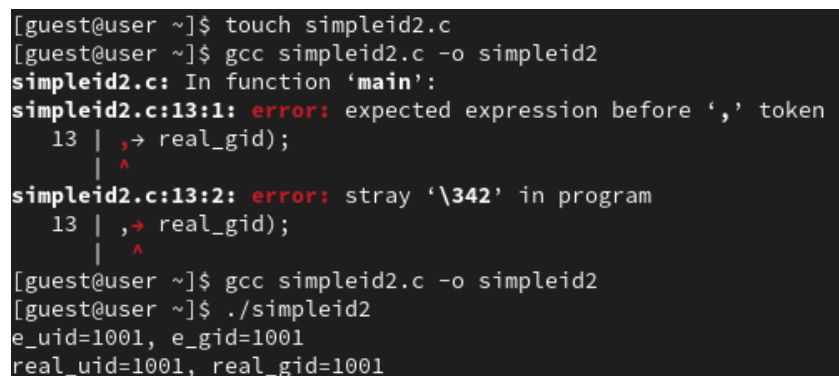
gid_t e_gid = getegid();      // Получение эффективного GID

// Вывод реальных и эффективных идентификаторов
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;
}

```

Получившуюся программу назвала simpleid2.c, скомпилировала и запустила simpleid2.c (рис. 4.3):



```

[guest@user ~]$ touch simpleid2.c
[guest@user ~]$ gcc simpleid2.c -o simpleid2
simpleid2.c: In function 'main':
simpleid2.c:13:1: error: expected expression before ',' token
   13 | ,→ real_gid);
      | ^
simpleid2.c:13:2: error: stray '\342' in program
   13 | ,→ real_gid);
      | ^
[guest@user ~]$ gcc simpleid2.c -o simpleid2
[guest@user ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001

```

Рис. 4.3: Задание 1.6-7

От имени суперпользователя выполнила команды

```

chown root:guest /home/guest/simpleid2
chmod u+s /home/guest/simpleid2

```

Команда chown меняет владельца файла, а chmod u+s устанавливает SetUID бит, что позволяет выполнять программу с правами владельца (root). Далее выполнила проверку правильности установки новых атрибутов и смены владельца файла simpleid2. Ну и запустила simpleid2 и id. Прodelала то же самое относительно SetGID-бита (рис. 4.4):

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

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

```
[guest@user ~]$ su
Password:
[root@user guest]# chown root:guest /home/guest/simpleid2
[root@user guest]# chmod u+s /home/guest/simpleid2
[root@user guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 17720 Oct  5 10:30 simpleid2
[root@user guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@user guest]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r
:unconfined_t:s0-s0:c0.c1023
[root@user guest]# chown root:guest /home/guest/simpleid2
[root@user guest]# chmod g+s /home/guest/simpleid2
[root@user guest]# ls -l simpleid2
-rwxr-sr-x. 1 root guest 17720 Oct  5 10:30 simpleid2
[root@user guest]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@user guest]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r
:unconfined_t:s0-s0:c0.c1023
```

Рис. 4.4: Задание 1.8-12

Создала программу readfile.c:

```
#include <fcntl.h>          // Для функции open()
#include <stdio.h>          // Для функции printf()
#include <sys/stat.h>       // Для работы с атрибутами файлов
#include <sys/types.h>      // Для использования типов данных
#include <unistd.h>         // Для функций read(), close()

int main (int argc, char* argv[]) {
    unsigned char buffer[16];    // Буфер для хранения прочитанных данных
    size_t bytes_read;           // Количество прочитанных байт
    int i;
    int fd = open(argv[1], O_RDONLY); // Открытие файла на чтение

    do {
        bytes_read = read(fd, buffer, sizeof(buffer)); // Чтение данных из файла
```

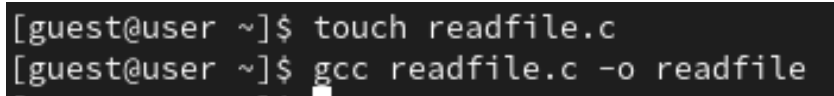
```

        for (i = 0; i < bytes_read; ++i) {
            printf("%c", buffer[i]); // Вывод прочитанных данных
        }
    } while (bytes_read == sizeof(buffer)); // Чтение продолжается до конца файла

    close(fd); // Закрывание файла
    return 0;
}

```

Откомпилировала её (рис. 4.5):



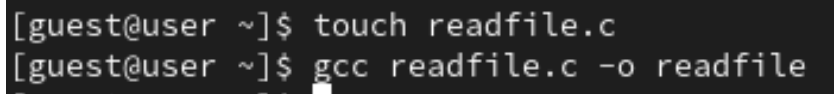
```

[guest@user ~]$ touch readfile.c
[guest@user ~]$ gcc readfile.c -o readfile

```

Рис. 4.5: Задание 1.13-14

Сменила владельца у файла readfile.c и изменила права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог (рис. 4.6):



```

[guest@user ~]$ touch readfile.c
[guest@user ~]$ gcc readfile.c -o readfile

```

Рис. 4.6: Задание 1.13-14

Проверила, что пользователь guest не может прочитать файл readfile.c, сменила у программы readfile владельца и установила SetU'D-бит. Проверила, может ли программа readfile прочитать файл readfile.c. Проверила, может ли программа readfile прочитать файл /etc/shadow? Ничего из выше перечисленного прочитать не смогли (рис. 4.7):

```
[guest@user ~]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@user ~]$ ./readfile readfile.c
bash: ./readfile: Permission denied
[guest@user ~]$ ./readfile /etc/shadow
bash: ./readfile: Permission denied
```

Рис. 4.7: Задание 1.16-19 (1)

Видим, что только суперпользователь может выполнить вышеперечисленное(рис. 4.8):

```

[root@user guest]# 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;
}
[root@user guest]# ./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;
}
[root@user guest]# ./readfile /etc/shadow
root:$6$5i0rpMKsyRp49od.$6leG0dcDz0MWfLjFi0NtCzkP5XB2WH4ntum71kn458ZC8FIH
oolPeowhWX/7EgcMzzRhd.t5m2ypirvlqEq.q.:0:99999:7:::
bin:!:19820:0:99999:7:::
daemon:!:19820:0:99999:7:::

```

Рис. 4.8: Задание 1.16-19 (2)

Начали исследование sticky-бита.

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:

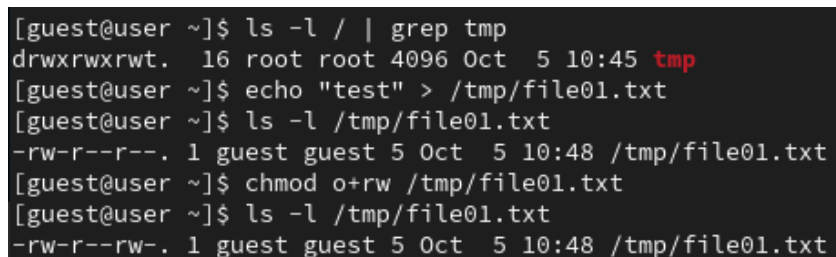

```
echo "test" > /tmp/file01.txt
```

Просмотрела атрибуты у только что созданного файла и разрешила чтение и запись для категории пользователей «все остальные» (рис. 4.9):

```
ls -l /tmp/file01.txt
chmod o+rw /tmp/file01.txt
ls -l /tmp/file01.txt
```

От пользователя guest2 (не являющегося владельцем) попробовала прочитать файл /tmp/file01.txt:

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



```
[guest@user ~]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 Oct  5 10:45 tmp
[guest@user ~]$ echo "test" > /tmp/file01.txt
[guest@user ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Oct  5 10:48 /tmp/file01.txt
[guest@user ~]$ chmod o+rw /tmp/file01.txt
[guest@user ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Oct  5 10:48 /tmp/file01.txt
```

Рис. 4.9: Задание 2.1-4

От пользователя guest2 попробовала дозаписать в файл /tmp/file01.txt слово test2. Однако мне этого не удалось сделать. От пользователя guest2 попробовала записать в файл /tmp/file01.txt слово test3, стеревав при этом всю имеющуюся в файле информацию, этого тоже не удалось сделать. От пользователя guest2 попробовала удалить файл /tmp/file01.txt, этого тоже не удалось сделать. После чего повысила свои права до суперпользователя и выполнила после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp:

```
chmod -t /tmp
```

Покинула режим суперпользователя. Повторила предыдущие шаги. Но сколько бы ни старалась, изменений не видно. Удалить файл от имени пользователя, не являющегося его владельцем не получилось (рис. 4.10):

```
[guest@user tmp]$ su guest2
Password:
[guest2@user tmp]$ cat /tmp/file01.txt
test
[guest2@user tmp]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@user tmp]$ cat /tmp/file01.txt
test
[guest2@user tmp]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@user tmp]$ cat /tmp/file01.txt
test
[guest2@user tmp]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory
[guest2@user tmp]$ su -
Password:
[root@user ~]# chmod -t /tmp
[root@user ~]# exit
logout
[guest2@user tmp]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 Oct  5 11:01 tmp
[guest2@user tmp]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@user tmp]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory
```

Рис. 4.10: Задание 2.5-14

После всего повысила свои права до суперпользователя и вернула атрибут `t` на директорию `/tmp`:

```
su -
chmod +t /tmp
exit
```

(рис. 4.11):

```
[guest2@user tmp]$ su -  
Password:  
[root@user ~]# chmod +t /tmp  
[root@user ~]# exit  
logout
```

Рис. 4.11: Задание 2.15

5 Выводы

Я изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получила практические навыки работы в консоли с дополнительными атрибутами. А также рассмотрела работу механизма смены идентификатора процессов пользователей, и влияние бита Sticky на запись и удаление файлов.

Список литературы

1. Права в Linux: команды и группы [Электронный ресурс]. Tprogger, 2023. URL: <https://tprogger.ru/articles/prava-v-linux-komandy-i-gruppy>.
2. Использование SETUID, SETGID и Sticky bit для расширенной настройки прав доступа в операционных системах Linux [Электронный ресурс]. RuVDS, 2024. URL: <https://ruvds.com/ru/helpcenter/suid-sgid-sticky-bit-linux/>.