

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ по лабораторной работе 5

ТЕМА «Дискреционное разграничение прав в Linux. Исследование влияния
дополнительных атрибутов»

по дисциплине «Информационная безопасность»

Выполнил/ла:

Студент/ка группы: НПИбд-02-21

Студенческий билет № 1032205421

Студент/ка: Стелина Петрити

Список содержания

[Список содержания.](#)

[Список изображений](#)

[Цель работы.](#)

[Последовательность выполнения работы](#)

[Выводы](#)

Список изображений

[рис. 1 создание simpleid.c](#)

[рис. 2 компиляция и id](#)

[рис. 3 Создание simpleid2.c](#)

[рис. 4 Компиляция и запуск simpleid2.c](#)

[рис. 5 команд chown,chmod](#)

[рис. 6 правильности установки атрибутов](#)

[рис. 7 запуск id](#)

[рис. 8 SetGID-бита](#)

[рис. 9 Создание readfile.c](#)

[рис. 10 Пункты 14,15,16](#)

[рис. 11 Пункты 17,18,19](#)

[рис. 12 Пункты 1,2,3](#)

[рис. 13 Пункты 4,5,6,7,8](#)

[рис. 14 Пункты 10,11](#)

[рис. 15 Нет атрибута](#)

[рис. 16 Возвращающий атрибут t](#)

[рис. 17 Проверка](#)

Цель работы

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

Последовательность выполнения работы

5.3.1. Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте программу simpleid.c:

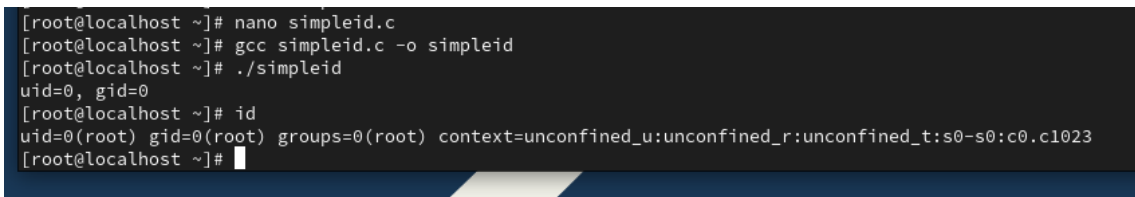


```
GNU nano 5.6.1 simpleid.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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

рис. 1 Создание 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.
Один и тот же uid и gid



```
[root@localhost ~]# nano simpleid.c
[root@localhost ~]# gcc simpleid.c -o simpleid
[root@localhost ~]# ./simpleid
uid=0, gid=0
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

рис. 2 компиляция и id

6. Усложните программу, добавив вывод действительных идентификаторов:
Получившуюся программу назовите simpleid2.c.

```
GNU nano 5.6.1 simpleid2.c
#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;
}
```

рис. 3 Создание *simpleid2.c*

7. Скомпилируйте и запустите *simpleid2.c*: `gcc simpleid2.c -o simpleid2, ./simpleid2`

```
[root@localhost ~]# nano simpleid2.c
[root@localhost ~]# gcc simpleid2.c -o simpleid2
[root@localhost ~]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

рис. 4 Компиляция и запуск *simpleid2.c*

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`.

```
[root@localhost ~]# sudo chown root:guest simpleid
[root@localhost ~]# chmod u+s simpleid
[root@localhost ~]# ls -l simpleid
-rwsr-xr-x. 1 root guest 17616 Oct  1 16:38 simpleid
[root@localhost ~]#
```

рис. 5 команд *chown, chmod*

Поясните, что делают эти команды.

sudo (SuperUser DO)

- **Что это?** `sudo` — это команда, которая позволяет пользователям выполнять команды с привилегиями суперпользователя (`root`) или другого пользователя, указанного в конфигурационном файле `/etc/sudoers`.

su (Substitute User)

- **Что это?** `su` — это команда, которая позволяет пользователю переключиться на другого пользователя, обычно на суперпользователя (`root`).

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

```
[root@localhost ~]# ls -l simpleid
-rwsr-xr-x. 1 root guest 17616 Oct  1 16:38 simpleid
[root@localhost ~]#
```

рис. 6 правильности установки атрибутов

11. Запустите simpleid2 и id: `./simpleid2`, `id` Сравните результаты.

```
[root@localhost ~]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

рис. 7 запуск, id

12. Прodelайте тоже самое относительно SetGID-бита.

```
[root@localhost ~]# chmod g+s simpleid
[root@localhost ~]# ls -l simpleid
ls: cannot access '-': No such file or directory
simpleid
[root@localhost ~]# ls -l simpleid
-rwsr-sr-x. 1 root guest 17616 Oct  1 16:38 simpleid
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

рис. 8 SetGID-бита

13. Создайте программу readfile.c:

```
root@localhost:~
GNU nano 5.6.1 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. 9 Создание 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.

```
[root@localhost ~]# gcc readfile.c -o readfile
[root@localhost ~]# chown root:root readfile.c
[root@localhost ~]# chmod 400 readfile.c
[root@localhost ~]# 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@localhost ~]#
```

рис. 10 Пункты 14,15,16

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? Да, читается.

```

[root@localhost ~]# chown root:root readfile
[root@localhost ~]# chmod u+s readfile
[root@localhost ~]# ./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@localhost ~]# ./readfile /etc/shadow
root:$6$2QMsiu9kwXfJxKH7$lcV0L9tKpFRuS3Kcl2/BegZyDhb2fG7j3/kXpf6SjRi1h6NNhm04wT9bUoeCyVE18yoRACZ/30
L80h/9r3e1::0:99999:7:::
bin:*:19820:0:99999:7:::
daemon:*:19820:0:99999:7:::
adm:*:19820:0:99999:7:::
lp:*:19820:0:99999:7:::

```

рис. 11 Пункты 17,18,19

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`

```

[root@localhost ~]# ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 Oct 1 17:33 tmp
[root@localhost ~]# echo "test" > /tmp/file01.txt
[root@localhost ~]# ls -l /tmp/file01.txt
-rw-r--r--. 1 root root 5 Oct 1 17:37 /tmp/file01.txt
[root@localhost ~]# chmod o+rw /tmp/file01.txt
[root@localhost ~]# ls -l /tmp/file01.txt
-rw-r--rw-. 1 root root 5 Oct 1 17:37 /tmp/file01.txt
[root@localhost ~]#

```

рис. 12 Пункты 1,2,3

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? Нет, операция была запрещена.

```
[root@localhost ~]# su - guest2
[guest2@localhost ~]$ cat /tmp/file01.txt
test
[guest2@localhost ~]$ echo "test3" > /tmp/file01.txt
[guest2@localhost ~]$ cat /tmp/file01.txt
test3
[guest2@localhost ~]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@localhost ~]$
```

рис. 13 Пункты 4,5,6,7,8

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`

```
[guest@localhost ~]$ su -
Password:
[root@localhost ~]# chmod -t /tmp
[root@localhost ~]# exit
logout
[guest@localhost ~]$
```

рис. 14 Пункты 10,11

12. От пользователя `guest2` проверьте, что атрибута `t` у директории `/tmp` нет: `ls -l / | grep tmp`

```
[guest2@localhost ~]$ ls -l / | grep tmp
drwxrwxrwx. 17 root root 4096 Oct  1 17:48 tmp
[guest2@localhost ~]$
```

рис. 15 Нет атрибута

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`

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


рис. 16 Возвращающий атрибут *t*

```
[guest2@localhost ~]$ ls -l / | grep tmp
drwxrwxrwt. 19 root root 4096 Oct  1 17:50 tmp
[guest2@localhost ~]$
```

рис. 17 Проверка

Вывод

В этой лабораторной работе мы изучили, как работают специальные атрибуты файлов в Linux. Мы увидели, как они помогают контролировать доступ к файлам и программам, а также защищают общие папки от удаления файлов другими пользователями.