

Лабораторная работа №5

Саргсян Арам Грачьевич

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Выводы	13
	Список литературы	14

Список иллюстраций

3.1	Создание файла	8
3.2	Редакция файла	8
3.3	Создание файла	9
3.4	Редакция файла	9
3.5	Редакция файла	9
3.6	Файл readfile	10
3.7	Изменение прав доступа	10
3.8	Изменение прав доступа	11
3.9	Пользователь guest	11
3.10	Пользователь guest2	11
3.11	Пользователь root	12

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 Теоретическое введение

Есть 3 вида разрешений. Они определяют права пользователя на 3 действия: чтение, запись и выполнение. В Linux эти действия обозначаются вот так:

- **r** — read (чтение) — право просматривать содержимое файла;
- **w** — write (запись) — право изменять содержимое файла;
- **x** — execute (выполнение) — право запускать файл, если это программа или скрипт.

У каждого файла есть 3 группы пользователей, для которых можно устанавливать права доступа.

- **owner** (владелец) — отдельный человек, который владеет файлом. Обычно это тот, кто создал файл, но владельцем можно сделать и кого-то другого.
- **group** (группа) — пользователи с общими заданными правами.
- **others** (другие) — все остальные пользователи, не относящиеся к группе и не являющиеся владельцами[1].

Чтобы увидеть текущие назначения владельца, нужно использовать команду `ls -l`. Эта команда показывает пользователя и группу-владельца.

Чтобы применить соответствующие разрешения, первое, что нужно учитывать, это владение. Для этого есть команда `chown`[2].

Для того, чтобы позволить обычным пользователям выполнять программы от имени суперпользователя без знания его пароля была придумана такая вещь, как SUID и SGID биты. Рассмотрим эти полномочия подробнее.

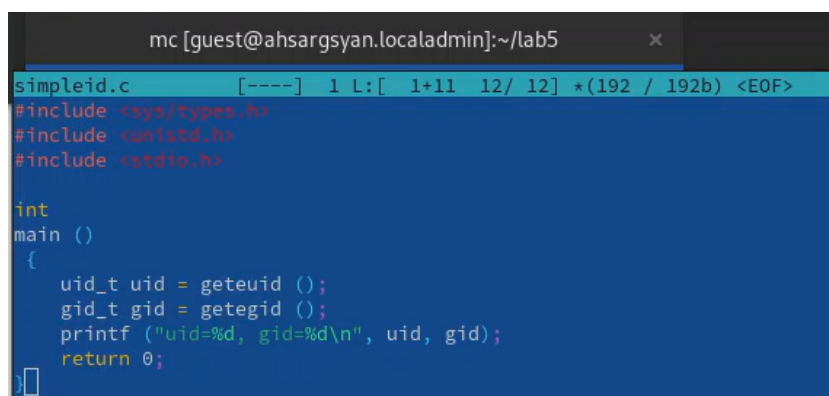
- **SUID** - если этот бит установлен, то при выполнении программы, id пользователя, от которого она запущена заменяется на id владельца файла. Фактически, это позволяет обычным пользователям запускать программы от имени суперпользователя;
- **SGID** - этот флаг работает аналогичным образом, только разница в том, что пользователь считается членом группы, с которой связан файл, а не групп, к которым он действительно принадлежит. Если SGID флаг установлен на каталог, все файлы, созданные в нем, будут связаны с группой каталога, а не пользователя. Такое поведение используется для организации общих папок;
- **Sticky-bit** - этот бит тоже используется для создания общих папок. Если он установлен, то пользователи могут только создавать, читать и выполнять файлы, но не могут удалять файлы, принадлежащие другим пользователям[3].

3 Выполнение лабораторной работы

1. От имени пользователя guest создал файл simpleid (рис. 3.1, 3.2).

```
[guest@ahsargsyan dir1]$  
[guest@ahsargsyan ~]$ mkdir lab5 && cd lab5  
[guest@ahsargsyan lab5]$ touch simpleid.c  
[guest@ahsargsyan lab5]$ gcc simpleid.c -o simpleid  
[guest@ahsargsyan lab5]$ ./simpleid  
uid=1001, gid=1001  
[guest@ahsargsyan lab5]$
```

Рис. 3.1: Создание файла



```
mc [guest@ahsargsyan.localadmin]:~/lab5  
simpleid.c [----] 1 L: [ 1+11 12/ 12] *(192 / 192b) <EOF>  
#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. 3.2: Редакция файла

2. Создал файл simpleid2 (рис. 3.3, 3.4).


```
5 |      uid_t e_uid = geteuid ();
6 |      }
7 |      }
[guest@ahsargsyan lab5]$ gcc simpleid2.c -o simpleid2 && ./simpleid2
e_id=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@ahsargsyan lab5]$ ls -l simpleid2
-rwxr-xr-x. 1 root guest 26064 Sep 24 12:50 simpleid2
[guest@ahsargsyan lab5]$ ./simpleid2
e_id=0, e_gid=1001
realuid=1001, real_gid=1001
[guest@ahsargsyan lab5]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@ahsargsyan lab5]$ touch readfile.c
[guest@ahsargsyan lab5]$ /bin/sh /var/tmp/mc-guest/mcusr0676B2
cc readfile.c -o readfile
/usr/bin/ld: /usr/lib/gcc/x86_64-redhat-linux/11/../../../../lib64/crt1.o: in function `_start':
(.text+0x1b): undefined reference to `main'
collect2: error: ld returned 1 exit status

[guest@ahsargsyan lab5]$ gcc readfile.c -o readfile
readfile.c:1:10: fatal error: fcntl.h: No such file or directory
1 | #include <fcntl.h>
  |
  |
compilation terminated.
[guest@ahsargsyan lab5]$ /bin/sh /var/tmp/mc-guest/mcusrE1YBC2
cc readfile.c -o readfile
readfile.c:1:10: fatal error: fcntl.h: No such file or directory
1 | #include <fcntl.h>
  |
  |
compilation terminated.

[guest@ahsargsyan lab5]$ gcc readfile.c -o readfile
readfile.c:1:10: fatal error: fcntl.h: No such file or directory
1 | #include <fcntl.h>
  |
  |
compilation terminated.
[guest@ahsargsyan lab5]$ gcc readfile.c -o readfile
readfile.c: In function 'main':
readfile.c:18:31: error: expected ';' before 'i'
18 |         for (i=0; i<bytes_read ++;) printf("%c", buffer[i]);
    |                               ^
    |
    |
[guest@ahsargsyan lab5]$ gcc readfile.c -o readfile
[guest@ahsargsyan lab5]$
```

Рис. 3.3: Создание файла

```
mc [guest@ahsargsyan.localadmin]:~/lab5 × root@ahsargsyan:/home/guest × ahsargsya
simpleid2.c [-M--] 9 L: [ 1+ 8 9/ 18] *(118 / 344b) 0032 0x020
#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_id=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
    ....

    return 0;
}
```

Рис. 3.4: Редакция файла

3. От имени суперпользователя выполнил нужные команды (рис. 3.5).

```
known: cannot access '/home/guest/simpleid2': No such file or directory
[root@ahsargsyan guest]# chown root:guest /home/guest/lab5/simpleid2
[root@ahsargsyan guest]# chmod u+s /home/guest/lab5/simpleid2
[root@ahsargsyan guest]#
```

Рис. 3.5: Редакция файла

4. Создал readfile.c, изменил права доступа (рис. 3.6, 3.7).

```
[guest@ahsargsyan lab5]$ gcc readfile.c -o readfile
[guest@ahsargsyan lab5]$ 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;
}[guest@ahsargsyan lab5]$
```

Рис. 3.6: Файл readfile

```
[root@ahsargsyan lab5]# ls
readfile readfile.c simpleid simpleid2 simpleid2.c simpleid.c
[root@ahsargsyan lab5]# chmod g-rw readfile.c
[root@ahsargsyan lab5]# chmod g-rw readfile
```

Рис. 3.7: Изменение прав доступа

5. Прочитал с помощью readfile etc/shadow (рис. 3.8).

4 Выводы

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

Список литературы

1. Права доступа в Linux [Электронный ресурс]. 2023. URL: <https://codechick.io/tutorials/unix-linux/unix-linux-permissions>.
2. Права в Linux (chown, chmod, SUID, GUID, sticky bit, ACL, umask) [Электронный ресурс]. 2023. URL: <https://habr.com/ru/articles/469667/>.
3. Права доступа к файлам в Linux [Электронный ресурс]. 2023. URL: <https://losst.pro/prava-dostupa-k-fajlam-v-linux>.