

## Прагматика выполнения лабораторной работы

---

Студенты должны разбираться в работе с атрибутами файлов и директорий, а также знать как пользователи с разными правами доступа взаимодействуют с ними. Все это необходимо для глубоко погружения в среду Centos и для повышения безопасности в системе.

## Цель выполнения лабораторной 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.

## Задачи выполнения лабораторной работы

---

1. Создать программу `simpleid.c`, скомпилировать и исполнить. Сравнить результат вывода программы с командой `id`.
2. Создать программу `simpleid2.c`, где добавить вывод действительных идентификаторов. Скомпилировать программу и исполнить. Сравнить результат вывода программы с командой `id`.
3. Изменить владельца скомпилированного файла `simpleid2` на `root` и поставить атрибут `u+s` на этот файл. Запустить файл `simpleid2` и сравнить с выводом `id`.
4. Установить SetGid-бит на файл `simpleid2`. Повторить прошлый пункт.
5. Создать программу `readfile.c`, откомпилировать. Поменять владельца у файла `readfile.c` на `root`, и поменять атрибуты так, чтобы пользователь `saat` не мог прочитать файл `readfile.c`.
6. Сменить у файла `readfile` владельца и установить `u+s`. Проверить может ли программа `readfile` прочитать файл `readfile.c`. Проверить, может ли программа `readfile` прочитать файл `/etc/shadow`.
7. Создать файл `file01.txt` в директории `/tmp`. Провести серию операций с файлом, где проверить возможность атрибута `t`, который стоит на директории `tmp`.
8. Снять атрибут `t` с каталога `tmp` и повторили все операции с фалом `file01.txt` из предыдущего пункта.

## Результаты выполнения лабораторной работы

---

1. Создала программу `simpleid.c`, скомпилировала.

(рис. -@fig:002)




```
#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;
}
```

Выполнила программу ./simpleid и id. Результаты они вывели одинаковые, так как обе программы просто выводят id пользователя, которому принадлежит файл и id группы файла.

(рис. -@fig:004)

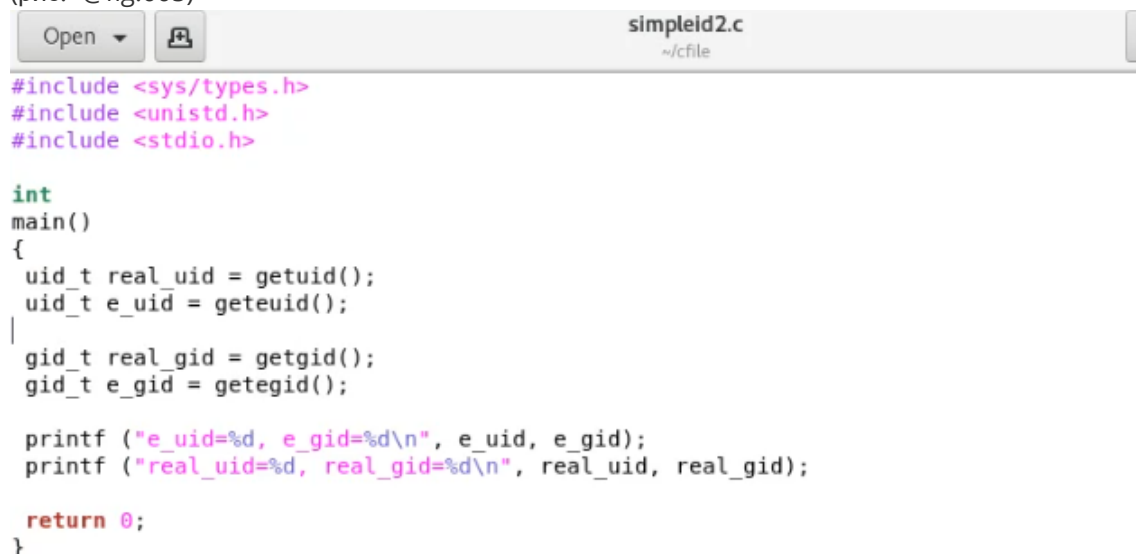


```
[caat@mpak cfile]$ gcc simpleid.c -o simpleid
[caat@mpak cfile]$ ls
simpleid simpleid.c
[caat@mpak cfile]$ ./simpleid
uid=1002, gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$
```

## Результаты выполнения лабораторной работы

2. Создала программу simpleid2.c, где добавила вывод действительных идентификаторов. Скомпилировала программу и исполнила.

(рис. -@fig:005)



```
#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_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. -@fig:006)

```
[caat@mpak cfile]$ ./simpleid
uid=1002, gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$ gcc simpleid2.c -o simpleid2
[caat@mpak cfile]$ ./simpleid2
e_uid=1002, e_gid=1003
real uid=1002, real gid=1003
[caat@mpak cfile]$
```

## Результаты выполнения лабораторной работы

3. Изменила владельца скомпилированного файла simpleid2 на root и поставила атрибут u+s на этот файл.

(рис. -@fig:009)

```
[caat@mpak cfile]$ ls -l simpleid2
-rwsrwxr-x. 1 root caat 17648 Nov 13 16:22 simpleid2
[caat@mpak cfile]$ ./simpleid2
e_uid=0, e_gid=1003
real uid=1002, real gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$
```

e\_uid показывает какие права пользователя использует программа. Программа пользуется правами суперюзера (думает, что программу запустил root). Это значение поменялось относительно последнего запуска, т.к. мы добавили права u+s. То же самое и с e\_gid. real\_uid указывает фактического пользователя, который запустил процесс. Запустил его caat. То же самое и с real\_gid. id показывает те же самые результаты, только нигде не показывает информацию о правах суперюзера.

## Результаты выполнения лабораторной работы

4. Установила SetGid-бит на файл simpleid2.

(рис. -@fig:011)

```
r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$ ./simpleid2
e_uid=0, e_gid=1003
real uid=1002, real gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$ ls -l simpleid2
-rwsrwsr-x. 1 root caat 17648 Nov 13 16:22 simpleid2
[caat@mpak cfile]$
```

Как можно увидеть на последней картинке, у файла действительно установились права g+s, поэтому теперь на месте X можно увидеть S. Команды снова вывели одинаковую непротиворечивую информацию.

## Результаты выполнения лабораторной работы

5. Создала программу readfile.c, откомпилировала. Поменяла владельца у файла readfile.c на root, и поменяла атрибуты так, чтобы пользователь caat не мог прочитать файл readfile.c.

(рис. -@fig:012)

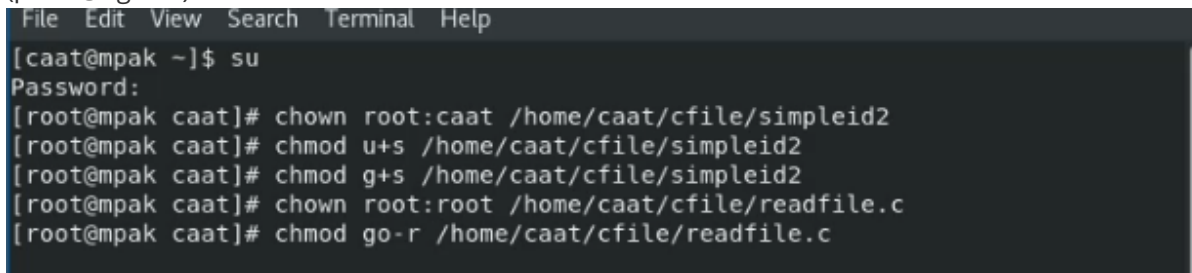


```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;

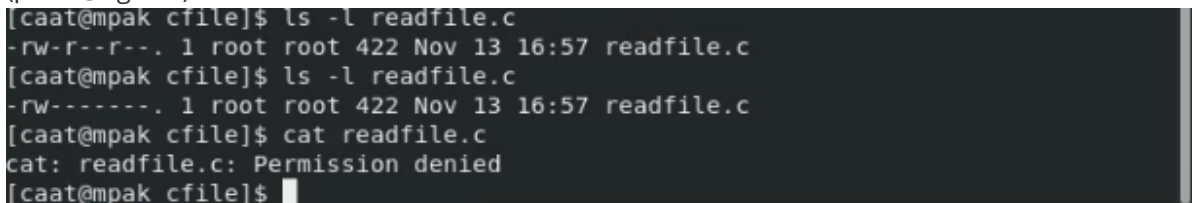
    int fd = open (argv[1], O_RDONLY);
    do
    {
        bytes_read = read (fd, buffer, sizeof(buffer));
        for(i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
    }
    while(bytes_read == sizeof(buffer));
    close (fd);
    return 0;
}
```

(рис. -@fig:014)



```
File Edit View Search Terminal Help
[caat@mpak ~]$ su
Password:
[root@mpak caat]# chown root:caat /home/caat/cfile/simpleid2
[root@mpak caat]# chmod u+s /home/caat/cfile/simpleid2
[root@mpak caat]# chmod g+s /home/caat/cfile/simpleid2
[root@mpak caat]# chown root:root /home/caat/cfile/readfile.c
[root@mpak caat]# chmod go-r /home/caat/cfile/readfile.c
```

(рис. -@fig:015)



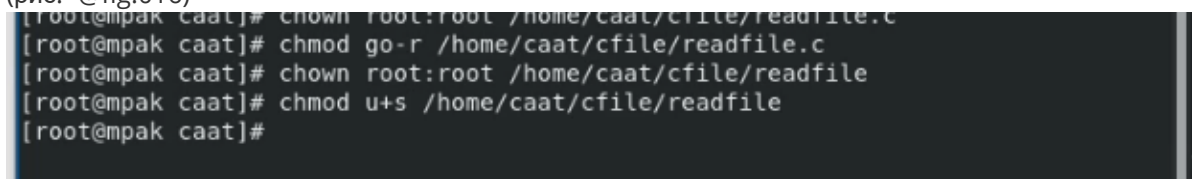
```
[caat@mpak cfile]$ ls -l readfile.c
-rw-r--r--. 1 root root 422 Nov 13 16:57 readfile.c
[caat@mpak cfile]$ ls -l readfile.c
-rw-----. 1 root root 422 Nov 13 16:57 readfile.c
[caat@mpak cfile]$ cat readfile.c
cat: readfile.c: Permission denied
[caat@mpak cfile]$
```

Теперь только пользователь root может читать этот файл, а пользователь caat получает отказ в операции чтение.

## Результаты выполнения лабораторной работы

6. Сменила у файла readfile владельца и установить u+s. Проверила может ли программа readfile прочитать файл readfile.c. Проверила, может ли программа readfile прочитать файл /etc/shadow.

(рис. -@fig:016)



```
[root@mpak caat]# chown root:root /home/caat/cfile/readfile.c
[root@mpak caat]# chmod go-r /home/caat/cfile/readfile.c
[root@mpak caat]# chown root:root /home/caat/cfile/readfile
[root@mpak caat]# chmod u+s /home/caat/cfile/readfile
[root@mpak caat]#
```

(рис. -@fig:017)

```
[caat@mpak cfile]$ ./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;
}
[caat@mpak cfile]$ ./readfile /etc/shadow
root:$6$NKVXPM5T1jrX.FXZ$KhYx2KDxA6qu0z/Uzp/sVf2n92DwG5HeUI3A6yvTm/00792bCUCWj0/TLy9MJ5ZYV8L48J.Yv1WJS.E
qN7ugE0::0:99999:7:::
bin:!:18397:0:99999:7:::
daemon:!:18397:0:99999:7:::
adm:!:18397:0:99999:7:::
lp:!:18397:0:99999:7:::
sync:!:18397:0:99999:7:::
shutdown:!:18397:0:99999:7:::
halt:!:18397:0:99999:7:::
mail:!:18397:0:99999:7:::
operator:!:18397:0:99999:7:::
games:!:18397:0:99999:7:::
ftp:!:18397:0:99999:7:::
nobody:!:18397:0:99999:7:::
dbus:!!:18899:::
systemd-coredump:!!:18899:::
systemd-resolve:!!:18899:::
tss:!!:18899:::
```

Так как мы дали нашему исполняемому файлу особый атрибут S, теперь этот файл будет пользоваться правами суперюзера. На несколько пунктов до этого мы изменили владельца файла readfile.c, т.е теперь его может читать только root. Так как наш скрипт readfile пользовался правами суперюзера, он и мог прочитать нужный нам файл readfile.c. Тоже самой и произошло с /etc/shadow, так как проводить операции с ним может только суперюзер.

## Результаты выполнения лабораторной работы

7. Создала файл file01.txt в директории /tmp. Провела серию операций с файлом, где проверяла возможность атрибута t, который стоит на директории tmp.

(рис. -@fig:020)

```
[root@mpak caat]# su guest
[guest@mpak caat]$ cat /tmp/file01.txt
tets
[guest@mpak caat]$ cat /tmp/file01.txt echo "tets2">/tmp/file01.txt
cat: echo: Permission denied
cat: tets2: Permission denied
[guest@mpak caat]$ echo "tets2">/tmp/file01.txt
[guest@mpak caat]$ cat /tmp/file01.txt
tets2
[guest@mpak caat]$ echo "tets3">>/tmp/file01.txt
[guest@mpak caat]$ cat /tmp/file01.txt
tets2
tets3
[guest@mpak caat]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest@mpak caat]$
```

Чтение файла работает.

Дозапись в файл работает.

Дозапись в файл со стиранием предыдущей информации - работает  
Файл не удалить удалось.

## Результаты выполнения лабораторной работы

8. Сняла атрибут t с каталога tmp и повторила все операции с файлом file01.txt из предыдущего пункта.

(рис. -@fig:021)

```
tmp: cannot remove /tmp/file01.txt: operation not permitted
[guest@mpak caat]$ su
Password:
[root@mpak caat]# cmod -t /tmp
bash: cmod: command not found...
Similar command is: 'kmod'
[root@mpak caat]# chmod -t /tmp
[root@mpak caat]# exit
exit
[guest@mpak caat]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 Nov 13 17:30 tmp
[guest@mpak caat]$
```

(рис. -@fig:022)

```
[guest@mpak caat]$ cat /tmp/file01.txt
tets2
tets3
[guest@mpak caat]$ echo "tets3">>/tmp/file01.txt
[guest@mpak caat]$ cat /tmp/file01.txt
tets2
tets3
tets3
[guest@mpak caat]$ echo "tets3">/tmp/file01.txt
[guest@mpak caat]$ cat /tmp/file01.txt
tets3
[guest@mpak caat]$ rm /tmp/file01.txt
[guest@mpak caat]$
```

Чтение файла работает.  
Дозапись в файл работает.  
Дозапись в файл со стиранием предыдущей информации - работает  
Файл удалить удалось.

## Результаты выполнения лабораторной работы

### Вывод

Изучила механизм изменения идентификаторов, применения 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.

### {.standout}

Спасибо за внимание