

# **Отчёт по лабораторной работе №5**

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

Выполнила: Павлова Полина Алексеевна,  
НПИбд-02-21, 1032212967

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	5.2.1. Подготовка лабораторного стенда . . . . .	7
3.2	5.3.1 Создание программы . . . . .	7
3.3	5.3.2. Исследование Sticky-бита . . . . .	12
<b>4</b>	<b>Вывод</b>	<b>15</b>
<b>5</b>	<b>Список литературы. Библиография</b>	<b>16</b>

## Список иллюстраций

3.1 (рис. 1. Установка gss) . . . . .	7
3.2 (рис. 2. simpleid.c) . . . . .	8
3.3 (рис. 3. 3-5 пункты задания лабораторной) . . . . .	8
3.4 (рис. 4. simpleid2.c) . . . . .	9
3.5 (рис. 5. 7 пункт задания лабораторной) . . . . .	9
3.6 (рис. 6. 8-12 пункты задания лабораторной) . . . . .	10
3.7 (рис. 7. readfile.c) . . . . .	10
3.8 (рис. 8. chmod) . . . . .	11
3.9 (рис. 9. 16-19 пункты Guest) . . . . .	11
3.10 (рис. 10. 16-18 пункты суперпользователь) . . . . .	12
3.11 (рис. 11. 19 пункт суперпользователь) . . . . .	12
3.12 (рис. 12. 1-3 пункты) . . . . .	13
3.13 (рис. 13. 4-12 пункты) . . . . .	14
3.14 (рис. 15. Возвращение атрибута) . . . . .	14

# 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

### 1. Дополнительные атрибуты файлов Linux

В Linux существует три основных вида прав — право на чтение (read), запись (write) и выполнение (execute), а также три категории пользователей, к которым они могут применяться — владелец файла (user), группа владельца (group) и все остальные (others). Но, кроме прав чтения, выполнения и записи, есть еще три дополнительных атрибута. [1]

- **Sticky bit**

Используется в основном для каталогов, чтобы защитить в них файлы. В такой каталог может писать любой пользователь. Но, из такой директории пользователь может удалить только те файлы, владельцем которых он является. Примером может служить директория /tmp, в которой запись открыта для всех пользователей, но нежелательно удаление чужих файлов.

- **SUID (Set User ID)**

Атрибут исполняемого файла, позволяющий запустить его с правами владельца. В Linux приложение запускается с правами пользователя, запустившего указанное приложение. Это обеспечивает дополнительную безопасность т.к. процесс с правами пользователя не сможет получить доступ к важным системным файлам, которые принадлежат пользователю root.

- **SGID (Set Group ID)**

Аналогичен `suid`, но относиться к группе. Если установить `sgid` для каталога, то все файлы созданные в нем, при запуске будут принимать идентификатор группы каталога, а не группы владельца, который создал файл в этом каталоге.

- **Обозначение атрибутов `sticky`, `suid`, `sgid`**

Специальные права используются довольно редко, поэтому при выводе программы `ls -l` символ, обозначающий указанные атрибуты, закрывает символ стандартных прав доступа.

Пример:

```
rwsrwsrwt
```

*где первая s — это `suid`, вторая s — это `sgid`, а последняя t — это `sticky bit`*

В приведенном примере не понятно, `gwt` — это `gw`- или `gwx`? Определить это просто. Если `t` маленькое, значит `x` установлен. Если `T` большое, значит `x` не установлен. То же самое правило распространяется и на `s`.

В числовом эквиваленте данные атрибуты определяются первым символом при четырехзначном обозначении (который часто опускается при назначении прав), например в правах `1777` — символ `1` обозначает `sticky bit`. Остальные атрибуты имеют следующие числовое соответствие:

1 — установлен `sticky bit`

2 — установлен `sgid`

4 — установлен `suid`

## **2. Компилятор GCC**

`GCC` - это свободно доступный оптимизирующий компилятор для языков `C`, `C++`. Собственно программа `gcc` это некоторая надстройка над группой компиляторов, которая способна анализировать имена файлов, передаваемые ей в качестве аргументов, и определять, какие действия необходимо выполнить. Файлы с расширением `.cc` или `.C` рассматриваются, как файлы на языке `C++`, файлы с расширением `.c` как программы на языке `C`, а файлы с расширением `.o` считаются объектными. [2]



```

#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: (рис. 2. 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.

```

[guest@papavlova12 ~]$ touch simpleid.c
[guest@papavlova12 ~]$ vi simpleid.c
[guest@papavlova12 ~]$ gcc simpleid.c -o simpleid
[guest@papavlova12 ~]$ ls
dir1 simpleid simpleid.c
[guest@papavlova12 ~]$ ./simpleid
bash: ./simpleid: команда не найдена...
[guest@papavlova12 ~]$ ./simpleid
uid=1001, gid=1001
[guest@papavlova12 ~]$ id
uid=1001(guest) gid=1001(guest) rpyнны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

```

Рис. 3.3: (рис. 3. 3-5 пункты задания лабораторной)

6. Усложните программу, добавив вывод действительных идентификаторов.



```

#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. 3.4: (рис. 4. simpleid2.c)

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

```

[guest@papavlova12 ~]$ vi simpleid2.c
[guest@papavlova12 ~]$ gcc simpleid2.c -o simpleid2
[guest@papavlova12 ~]$ ls
dir1  simpleid  simpleid2  simpleid2.c  simpleid.c
[guest@papavlova12 ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@papavlova12 ~]$

```

Рис. 3.5: (рис. 5. 7 пункт задания лабораторной)

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.

От имени суперпользователя выполнила команды “`sudo chown root:guest`  
`/home/guest/simpleid2`” и “`sudo chmod u+s /home/guest/simpleid2`”, затем  
 выполнила проверку правильности установки новых атрибутов и смены

владельца файла simpleid2 командой “sudo ls -l /home/guest/simpleid2” (рис. 3.9). Этими командами была произведена смена пользователя файла на root и установлен SetUID-бит.

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.

```
[guest@papavlova12 ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 17656 Dec 9 09:07 simpleid2
[guest@papavlova12 ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@papavlova12 ~]$ id
uid=1001(guest) gid=1001(guest) rpymu=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@papavlova12 ~]$ touch readfile.c
[guest@papavlova12 ~]$ vi readfile.c
[guest@papavlova12 ~]$ gcc readfile.c -o readfile
[guest@papavlova12 ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
```

Рис. 3.6: (рис. 6. 8-12 пункты задания лабораторной)

13. Создайте программу readfile.c
14. Откомпилируйте её. gcc readfile.c -o readfile

```
#include <stdio.h>
#include <stdlib.h>

int main(){
FILE *file = fopen("readfile.c","r");
if(file == NULL){
perror("error opening file");
return EXIT_FAILURE;
}
char line [256];
while(fgets(line,sizeof(line),file)){
printf("%s",line);
}
fclose(file);
}
```

Рис. 3.7: (рис. 7. readfile.c)

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.

```
[root@papavlova12 ~]# hown root:guest /home/guest/simpleid2
bash: hown: команда не найдена...
^[A[root@papavlova12 ~]# chown root:guest /home/guest/simpleid2
[root@papavlova12 ~]# chmod u+s /home/guest/simpleid2
[root@papavlova12 ~]# ls -l simpleid2
ls: невозможно получить доступ к 'simpleid2': Нет такого файла или каталога
[root@papavlova12 ~]# chown root:root /home/guest/readfile.c
[root@papavlova12 ~]# chmod 600 /home/guest/readfile.c
[root@papavlova12 ~]# chown root:root /home/guest/readfile
[root@papavlova12 ~]# chmod u+s /home/guest/readfile
[root@papavlova12 ~]# ./ /home/guest/readfile
-bash: ./: Это каталог
```

Рис. 3.8: (рис. 8. chmod)

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.

```
[guest@papavlova12 ~]# vi readfile.c
[guest@papavlova12 ~]$ gcc readfile.c -o readfile
[guest@papavlova12 ~]$ ./readfile
error opening file: Permission denied
[guest@papavlova12 ~]$ sudo ./readfile
```

Рис. 3.9: (рис. 9. 16-19 пункты Guest)

От имени суперпользователя все команды удастся выполнить.

```

[root@papavloval2 ~]# chown root:guest /home/guest/simpleid2
[root@papavloval2 ~]# ls -l simpleid2
ls: невозможно получить доступ к 'simpleid2': Нет такого файла или каталога
[root@papavloval2 ~]# chown root:root /home/guest/readfile.c
[root@papavloval2 ~]# chmod 600 /home/guest/readfile.c
[root@papavloval2 ~]# chown root:root /home/guest/readfile
[root@papavloval2 ~]# chmod u+s /home/guest/readfile
[root@papavloval2 ~]# ./ /home/guest/readfile
-bash: ./: Это каталог
[root@papavloval2 ~]# cd /home/guest/readfile
-bash: cd: /home/guest/readfile: Это не каталог
[root@papavloval2 ~]# cd /home/guest/read

```

Рис. 3.10: (рис. 10. 16-18 пункты суперпользователь)

```

games::19760:0:99999:7::
ftp::19760:0:99999:7::
nobody::19760:0:99999:7::
tss::19974:1::
systemd-coredump::19974:1::
dbus::19974:1::
polkitd::19974:1::
avahi::19974:1::
geoclue::19974:1::
cockpit-wsinstance::19974:1::
rtkit::19974:1::
staprunpriv::19974:1::
libstoragemgmt::19974:1::
colord::19974:1::
sssd::19974:1::
clevis::19974:1::
setroubleshoot::19974:1::
pipewire::19974:1::
flatpak::19974:1::
gdm::19974:1::
gnome-initial-setup::19974:1::
pesign::19974:1::
chrony::19974:1::
sshd::19974:1::
dnsmasq::19974:1::
tcpdump::19974:1::
papavloval2::0:99999:7::
guest:$6$rounds=100000$50JC17W..WuNoQ1Q5iFaNg1vWd1fu8sKGd8fuaghwCUyRVumadh99vXy.ug5bcg5zR6dtyX2mn0c
guest2:$6$rounds=100000$ULhUXK6cu8ZBAaAF$MKgFhn73n9Y6vWRNkaaQx/5Pm1JjxTaeyy97H.mqp8PtP5FMhvodXUtcTg
vboxadd::19974:1::
[root@papavloval2 ~]#

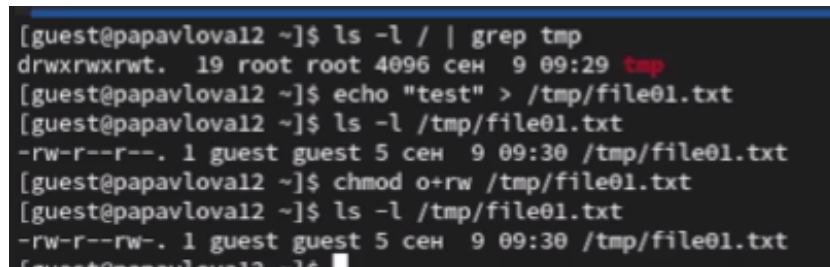
```

Рис. 3.11: (рис. 11. 19 пункт суперпользователь)

### 3.3 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`



```
[guest@papavlova12 ~]$ ls -l / | grep tmp
drwxrwxrwt. 19 root root 4096 сен  9 09:29 tmp
[guest@papavlova12 ~]$ echo "test" > /tmp/file01.txt
[guest@papavlova12 ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 сен  9 09:30 /tmp/file01.txt
[guest@papavlova12 ~]$ chmod o+rw /tmp/file01.txt
[guest@papavlova12 ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 сен  9 09:30 /tmp/file01.txt
```

Рис. 3.12: (рис. 12. 1-3 пункты)

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? Нет.

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? Нет.

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.

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`

```
Пароль:
[guest2@papavlova12 ~]$ cat /tmp/file01.txt
test
[guest2@papavlova12 ~]$ echo "test2" >> /tmp/file01.txt
-bash: /tmp/file01.txt: Отказано в доступе
[guest2@papavlova12 ~]$ echo "test2" > /tmp/file01.txt
-bash: /tmp/file01.txt: Отказано в доступе
[guest2@papavlova12 ~]$ cat /tmp/file01.txt
test
[guest2@papavlova12 ~]$ rm /tmp/file01.txt
rm: удалить защищённый от записи обычный файл '/tmp/file01.txt'? y
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена
[guest2@papavlova12 ~]$ su -
Пароль:
[root@papavlova12 ~]# chmod -t /tmp
[root@papavlova12 ~]# exit
выход
[guest2@papavlova12 ~]$ ls -l / | grep tmp
drwxrwxrwx. 19 root root 4096 сен  9 09:33 tmp
```

Рис. 3.13: (рис. 13. 4-12 пункты)

13. Повторите предыдущие шаги. Какие наблюдаются изменения?

При повторении всё получилось.

14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Удалось.
15. Повысьте свои права до суперпользователя и верните атрибут `t` на директорию `/tmp`: `su chmod +t /tmp exit`

```
[guest2@papavlova12 ~]$ su -
Пароль:
[root@papavlova12 ~]# chmod +t /tmp
[root@papavlova12 ~]# exit
выход
[guest2@papavlova12 ~]$ █
```

Рис. 3.14: (рис. 15. Возвращение атрибута)

## 4 Вывод

Были изучены механизмы изменения идентификаторов и применения SetUID- и Sticky-битов. Получены практические навыки работы в консоли с дополнительными атрибутами. Были рассмотрены работа механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов

## **5 Список литературы. Библиография**

[0] Методические материалы курса

[1] Дополнительные атрибуты: <https://tokmakov.msk.ru/blog/item/141>

[2] Компилятор GSS: <http://parallel.imm.uran.ru/freesoft/make/instrum.html>