

# **Отчёт по лабораторной работе №5**

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

Одеджими Олуваколаде НПИбд-01-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
2.1	Подготовка . . . . .	5
2.2	Изучение механики SetUID . . . . .	6
2.3	Исследование Sticky-бита . . . . .	11
<b>3</b>	<b>Выводы</b>	<b>14</b>
	<b>Список литературы</b>	<b>15</b>

# List of Figures

2.1	подготовка к работе . . . . .	6
2.2	программа simpleid . . . . .	7
2.3	результат программы simpleid . . . . .	7
2.4	программа simpleid2 . . . . .	8
2.5	результат программы simpleid2 . . . . .	9
2.6	программа readfile . . . . .	10
2.7	результат программы readfile . . . . .	11
2.8	исследование Sticky-бита . . . . .	13

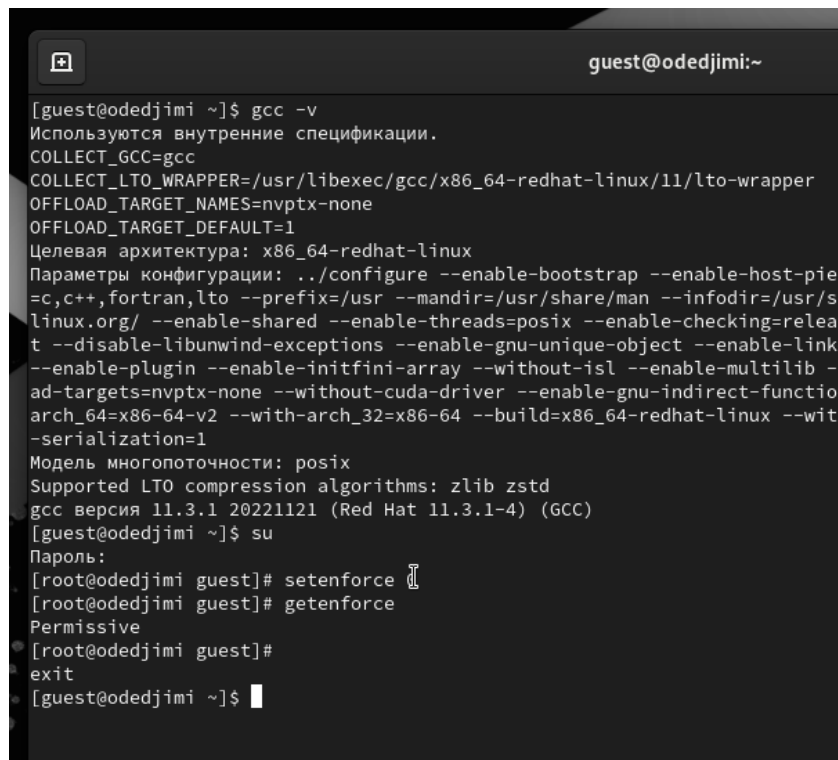
# 1 Цель работы

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 Выполнение лабораторной работы

### 2.1 Подготовка

1. Для выполнения части заданий требуются средства разработки приложений. Проверили наличие установленного компилятора gcc командой `gcc -v`:  
компилятор обнаружен.
2. Чтобы система защиты SELinux не мешала выполнению заданий работы, отключили систему запретов до очередной перезагрузки системы командой `setenforce 0`:
3. Команда `getenforce` вывела `Permissive`:

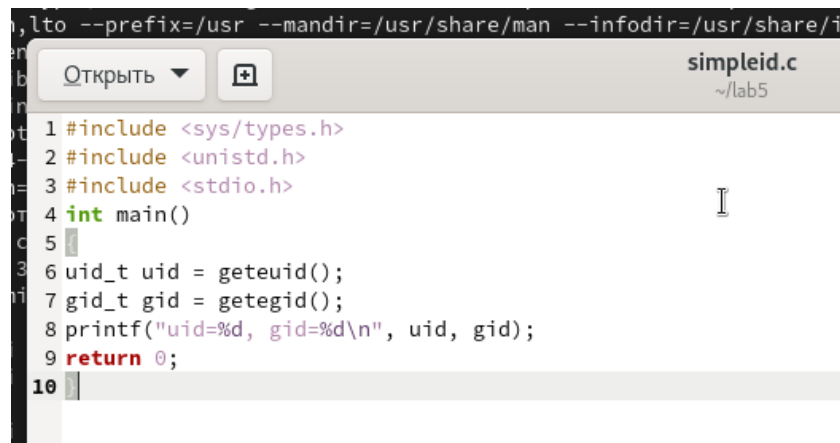
A terminal window with a dark background and light text. The window title is 'guest@odedjimi:~'. The user 'guest' runs 'gcc -v', which outputs detailed GCC version information (11.3.1) and configuration parameters. Then, the user runs 'su' to become root. The root user runs 'setenforce 0' and 'getenforce', which returns 'Permissive'. Finally, the user runs 'exit' to return to the 'guest' shell.

```
guest@odedjimi:~$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-host-pie
--enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share
linux.org/ --enable-shared --enable-threads=posix --enable-checking=release
--disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker
--enable-plugin --enable-initfini-array --without-isl --enable-multilib --
ad-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function
arch_64=x86-64-v2 --with-arch_32=x86-64 --build=x86_64-redhat-linux --with
--serialization=1
Модель многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 11.3.1 20221121 (Red Hat 11.3.1-4) (GCC)
guest@odedjimi ~]$ su
Пароль:
[root@odedjimi guest]# setenforce 0
[root@odedjimi guest]# getenforce
Permissive
[root@odedjimi guest]#
exit
guest@odedjimi ~]$
```

Figure 2.1: подготовка к работе

## 2.2 Изучение механики SetUID

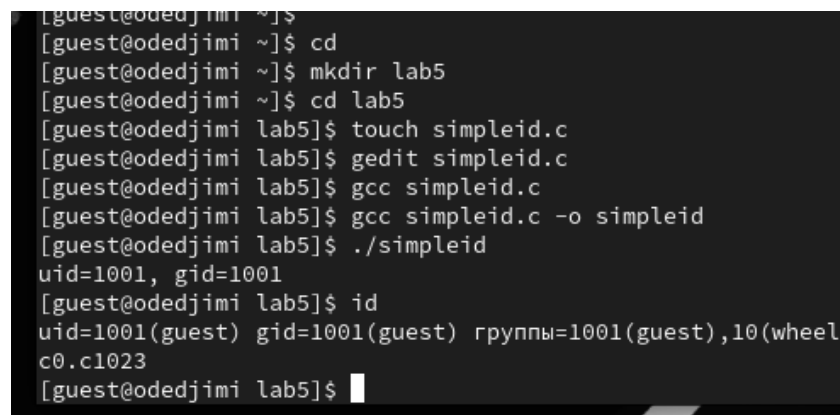
1. Вошли в систему от имени пользователя guest.
2. Написали программу simpleid.c.



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int main()
5 {
6     uid_t uid = geteuid();
7     gid_t gid = getegid();
8     printf("uid=%d, gid=%d\n", uid, gid);
9     return 0;
10 }
```

Figure 2.2: программа simpleid

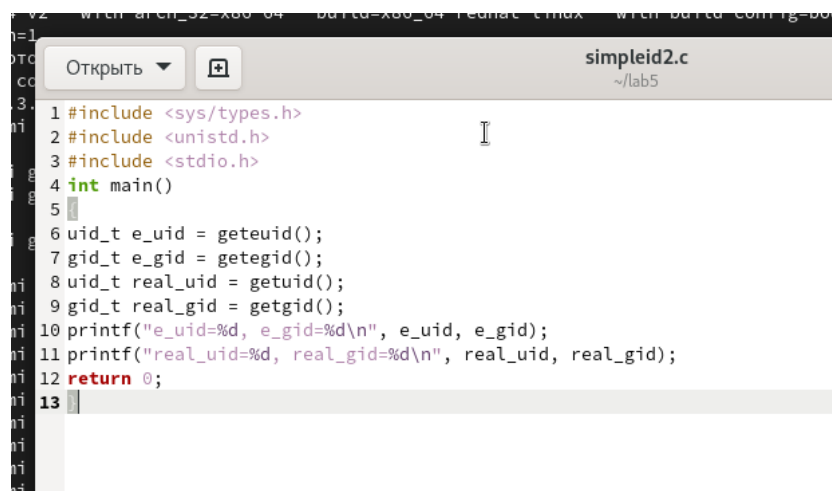
3. Скомпилировали программу и убедились, что файл программы создан: `gcc simpleid.c -o simpleid`
4. Выполнили программу `simpleid` командой `./simpleid`
5. Выполнили системную программу `id` с помощью команды `id`. `uid` и `gid` совпадает в обеих программах



```
[guest@odedjimi ~]$
[guest@odedjimi ~]$ cd
[guest@odedjimi ~]$ mkdir lab5
[guest@odedjimi ~]$ cd lab5
[guest@odedjimi lab5]$ touch simpleid.c
[guest@odedjimi lab5]$ gedit simpleid.c
[guest@odedjimi lab5]$ gcc simpleid.c
[guest@odedjimi lab5]$ gcc simpleid.c -o simpleid
[guest@odedjimi lab5]$ ./simpleid
uid=1001, gid=1001
[guest@odedjimi lab5]$ id
uid=1001(guest) gid=1001(guest) rpyны=1001(guest),10(wheel)
c0.c1023
[guest@odedjimi lab5]$
```

Figure 2.3: результат программы simpleid

6. Усложнили программу, добавив вывод действительных идентификаторов.



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int main()
5 {
6     uid_t e_uid = getuid();
7     gid_t e_gid = getegid();
8     uid_t real_uid = getuid();
9     gid_t real_gid = getgid();
10    printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
11    printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
12    return 0;
13 }
```

Figure 2.4: программа simpleid2

7. Скомпилировали и запустили simpleid2.c:

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

8. От имени суперпользователя выполнили команды:

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

9. Использовали su для повышения прав до суперпользователя

10. Выполнили проверку правильности установки новых атрибутов и смены владельца файла simpleid2:

```
ls -l simpleid2
```

11. Запустили simpleid2 и id:

```
./simpleid2
id
```

Результат выполнения программ теперь немного отличается

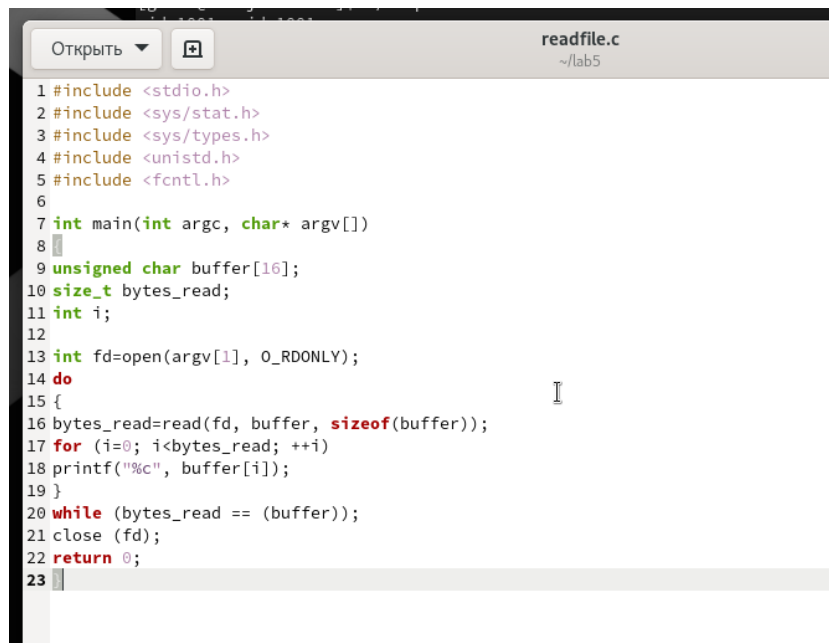


12. Проделали тоже самое относительно SetGID-бита.

```
[guest@odedjimi lab5]$  
[guest@odedjimi lab5]$ touch simpleid2.c  
[guest@odedjimi lab5]$ gedit simpleid2.c  
[guest@odedjimi lab5]$ gcc simpleid2.c  
[guest@odedjimi lab5]$ gcc simpleid2.c -o simpleid2  
[guest@odedjimi lab5]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@odedjimi lab5]$ su  
Пароль:  
[root@odedjimi lab5]# chown root:guest simpleid2  
[root@odedjimi lab5]# chmod u+s simpleid2  
[root@odedjimi lab5]# ./simpleid2  
e_uid=0, e_gid=0  
real_uid=0, real_gid=0  
[root@odedjimi lab5]# id  
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:  
[root@odedjimi lab5]# chmod g+s simpleid2  
[root@odedjimi lab5]# ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=0, real_gid=0  
[root@odedjimi lab5]#  
exit  
[guest@odedjimi lab5]$ ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@odedjimi lab5]$
```

Figure 2.5: результат программы simpleid2

13. Написали программу readfile.c



```
1 #include <stdio.h>
2 #include <sys/stat.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <fcntl.h>
6
7 int main(int argc, char* argv[])
8 {
9     unsigned char buffer[16];
10    size_t bytes_read;
11    int i;
12
13    int fd=open(argv[1], O_RDONLY);
14    do
15    {
16        bytes_read=read(fd, buffer, sizeof(buffer));
17        for (i=0; i<bytes_read; ++i)
18            printf("%c", buffer[i]);
19    }
20    while (bytes_read == (buffer));
21    close (fd);
22    return 0;
23 }
```

Figure 2.6: программа readfile

14. Откомпилировали её.

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

15. Сменили владельца у файла readfile.c и изменили права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

```
chown root:guest /home/guest/readfile.c
```

```
chmod 700 /home/guest/readfile.c
```

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

17. Сменили у программы readfile владельца и установили SetU'D-бит.

18. Проверили, может ли программа readfile прочитать файл readfile.c

19. Проверили, может ли программа readfile прочитать файл /etc/shadow

```

real_uid=1001, real_gid=1001
[guest@odedjimi lab5]$
[guest@odedjimi lab5]$
[guest@odedjimi lab5]$ touch readfile.c
[guest@odedjimi lab5]$ gedit readfile.c
[guest@odedjimi lab5]$ gcc readfile.c
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
    20 | while (bytes_read == (buffer));
        |                   ^~
[guest@odedjimi lab5]$ gcc readfile.c -o readfile
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
    20 | while (bytes_read == (buffer));
        |                   ^~
[guest@odedjimi lab5]$ su
Пароль:
[root@odedjimi lab5]# chown root:root readfile
[root@odedjimi lab5]# chmod -rwx readfile.c
[root@odedjimi lab5]# chmod u+s readfile
[root@odedjimi lab5]#
exit
[guest@odedjimi lab5]$ cat readfile

```

Figure 2.7: результат программы readfile

## 2.3 Исследование Sticky-бита

1. Выяснили, установлен ли атрибут Sticky на директории /tmp:

```
ls -l / | grep tmp
```

2. От имени пользователя `guest` создали файл `file01.txt` в директории `/tmp` со словом `test`:

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

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

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

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

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

Первоначально все группы имели право на чтение, а запись могли осуществлять все, кроме «остальных пользователей».

4. От пользователя (не являющегося владельцем) попробовали прочитать файл /file01.txt:

```
cat /file01.txt
```

5. От пользователя попробовали дозаписать в файл /file01.txt слово test3 командой:

```
echo "test2" >> /file01.txt
```

6. Проверили содержимое файла командой:

```
cat /file01.txt
```

В файле теперь записано:

```
Test
```

```
Test2
```

7. От пользователя попробовали записать в файл /tmp/file01.txt слово test4, стерев при этом всю имеющуюся в файле информацию командой. Для этого воспользовалась командой `echo "test3" > /tmp/file01.txt`

8. Проверили содержимое файла командой

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

9. От пользователя попробовали удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`, однако получила отказ.

10. От суперпользователя командой выполнили команду, снимающую атрибут t (Sticky-бит) с директории /tmp:

```
chmod -t /tmp
```

Покинули режим суперпользователя командой `exit`.

11. От пользователя проверили, что атрибута `t` у директории `/tmp` нет:

```
ls -l / | grep tmp
```

12. Повторили предыдущие шаги. Получилось удалить файл

13. Удалось удалить файл от имени пользователя, не являющегося его владельцем.

14. Повысили свои права до суперпользователя и вернули атрибут `t` на директорию `/tmp` :

```
su
```

```
chmod +t /tmp
```

```
exit
```

```
[guest@odedjimi lab5]$  
[guest@odedjimi lab5]$  
[guest@odedjimi lab5]$ ./readfile readfile.c  
#include <stdio.h>[guest@odedjimi lab5]$  
[guest@odedjimi lab5]$ ./readfile /etc/shadow  
root:$6$0mJpkglj[guest@odedjimi lab5]$  
[guest@odedjimi lab5]$  
[guest@odedjimi lab5]$  
[guest@odedjimi lab5]$ cd /tmp  
[guest@odedjimi tmp]$ echo test >> file01.txt  
[guest@odedjimi tmp]$ chmod g+rw file01.txt  
[guest@odedjimi tmp]$ su guest2  
Пароль:  
[guest2@odedjimi tmp]$ echo test2 >> file01.txt  
[guest2@odedjimi tmp]$ cat file01.txt  
test  
test2  
[guest2@odedjimi tmp]$ echo test > file01.txt  
[guest2@odedjimi tmp]$ rm file01.txt  
rm: невозможно удалить 'file01.txt': Операция не позволена  
[guest2@odedjimi tmp]$ su  
Пароль:  
[root@odedjimi tmp]# chmod -t /tmp  
[root@odedjimi tmp]#  
exit  
[guest2@odedjimi tmp]$ rm file01.txt  
[guest2@odedjimi tmp]$
```

Figure 2.8: исследование Sticky-бита

## 3 Выводы

Изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Также мы рассмотрели работу механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов.

# Список литературы

1. КОМАНДА CHATTR В LINUX
2. chattr