

# **Отчёт по лабораторной работе №5**

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

Солодовников Игорь НБИ-01-19

# 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 . . . . .	10
<b>3</b>	<b>Выводы</b>	<b>13</b>
	<b>Список литературы</b>	<b>14</b>

# List of Figures

2.1	подготовка к работе . . . . .	5
2.2	программа simpleid . . . . .	6
2.3	результат программы simpleid . . . . .	6
2.4	программа simpleid2 . . . . .	7
2.5	результат программы simpleid2 . . . . .	8
2.6	программа readfile . . . . .	8
2.7	результат программы readfile . . . . .	9
2.8	исследование Sticky-бита . . . . .	12

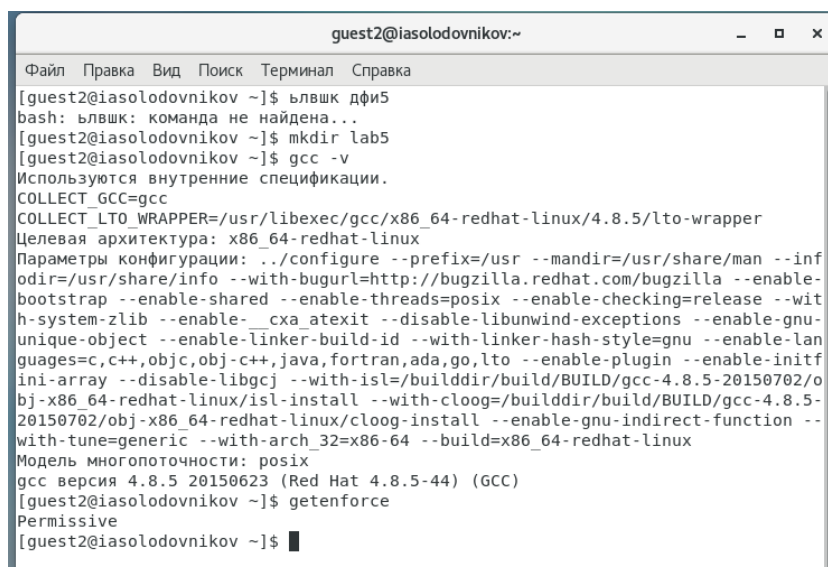
# 1 Цель работы

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`:

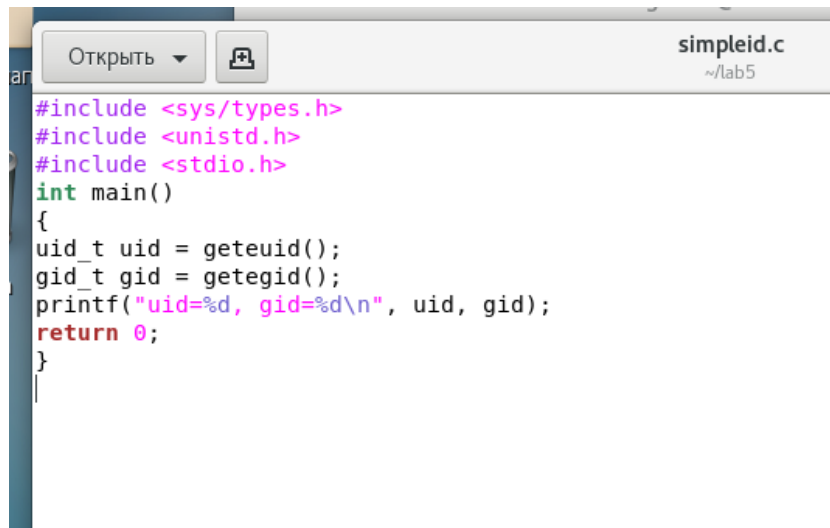


```
guest2@iasolodovnikov:~  
[guest2@iasolodovnikov ~]$ ьлвшк дфи5  
bash: ьлвшк: команда не найдена...  
[guest2@iasolodovnikov ~]$ mkdir lab5  
[guest2@iasolodovnikov ~]$ gcc -v  
Используются внутренние спецификации.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-wrapper  
Целевая архитектура: x86_64-redhat-linux  
Параметры конфигурации: ../configure --prefix=/usr --mandir=/usr/share/man --inf  
odir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-  
bootstrap --enable-shared --enable-threads=posix --enable-checking=release --wit  
h-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-  
unique-object --enable-linker-build-id --with-linker-hash-style=gnu --enable-lan  
guages=c,c++,objc,obj-c++,java,fortran,ada,go,lto --enable-plugin --enable-initf  
ini-array --disable-libgcj --with-isl=/builddir/build/BUILD/gcc-4.8.5-20150702/o  
bj-x86_64-redhat-linux/isl-install --with-cloog=/builddir/build/BUILD/gcc-4.8.5-  
20150702/obj-x86_64-redhat-linux/cloog-install --enable-gnu-indirect-function --  
with-tune=generic --with-arch_32=x86-64 --build=x86_64-redhat-linux  
Модель многопоточности: posix  
gcc версия 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)  
[guest2@iasolodovnikov ~]$ getenforce  
Permissive  
[guest2@iasolodovnikov ~]$
```

Figure 2.1: подготовка к работе

## 2.2 Изучение механики SetUID

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

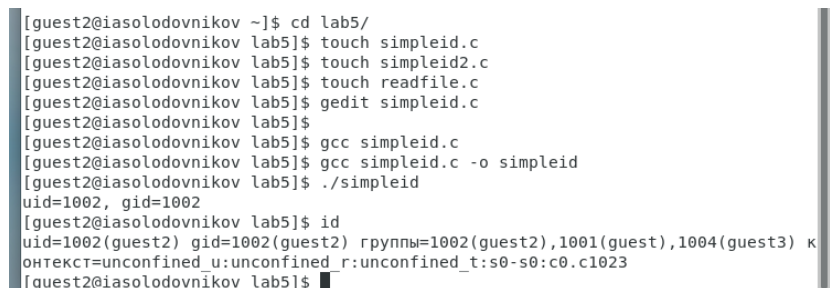


```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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

Figure 2.2: программа simpleid

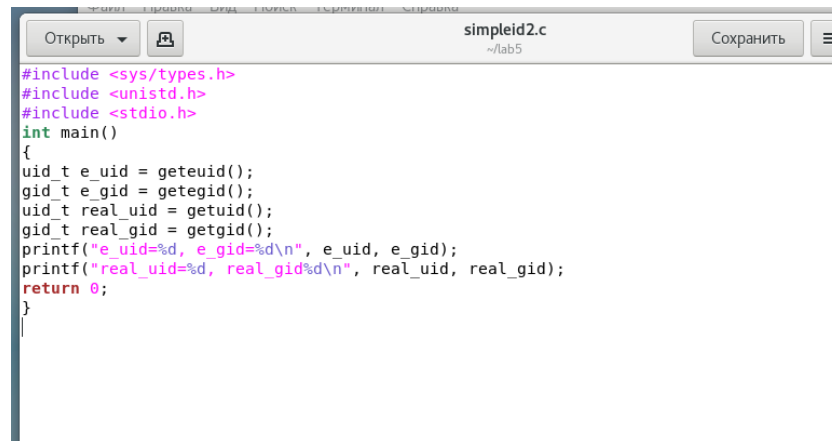
3. Скомпилировали программу и убедились, что файл программы создан: gcc simpleid.c -o simpleid
4. Выполнили программу simpleid командой ./simpleid
5. Выполнили системную программу id с помощью команды id. uid и gid совпадает в обеих программах



```
[guest2@iasolodovnikov ~]$ cd lab5/
[guest2@iasolodovnikov lab5]$ touch simpleid.c
[guest2@iasolodovnikov lab5]$ touch simpleid2.c
[guest2@iasolodovnikov lab5]$ touch readfile.c
[guest2@iasolodovnikov lab5]$ gedit simpleid.c
[guest2@iasolodovnikov lab5]$
[guest2@iasolodovnikov lab5]$ gcc simpleid.c
[guest2@iasolodovnikov lab5]$ gcc simpleid.c -o simpleid
[guest2@iasolodovnikov lab5]$ ./simpleid
uid=1002, gid=1002
[guest2@iasolodovnikov lab5]$ id
uid=1002(guest2) gid=1002(guest2) rpyнны=1002(guest2),1001(guest),1004(guest3) к
онтекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest2@iasolodovnikov lab5]$
```

Figure 2.3: результат программы simpleid

6. Усложнили программу, добавив вывод действительных идентификаторов.



```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
    uid_t e_uid = geteuid();
    gid_t e_gid = getegid();
    uid_t real_uid = getuid();
    gid_t real_gid = getgid();
    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;
}
```

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:

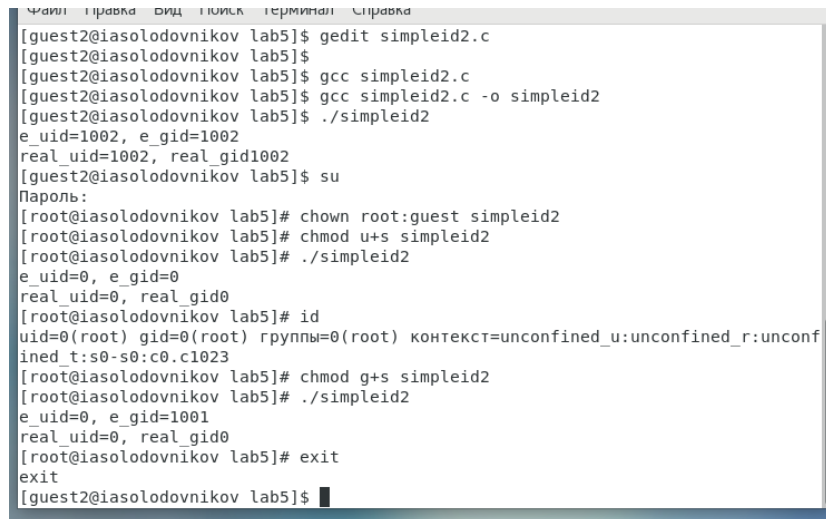
```
ls -l simpleid2
```

11. Запустили simpleid2 и id:

```
./simpleid2
id
```

Результат выполнения программ теперь немного отличается

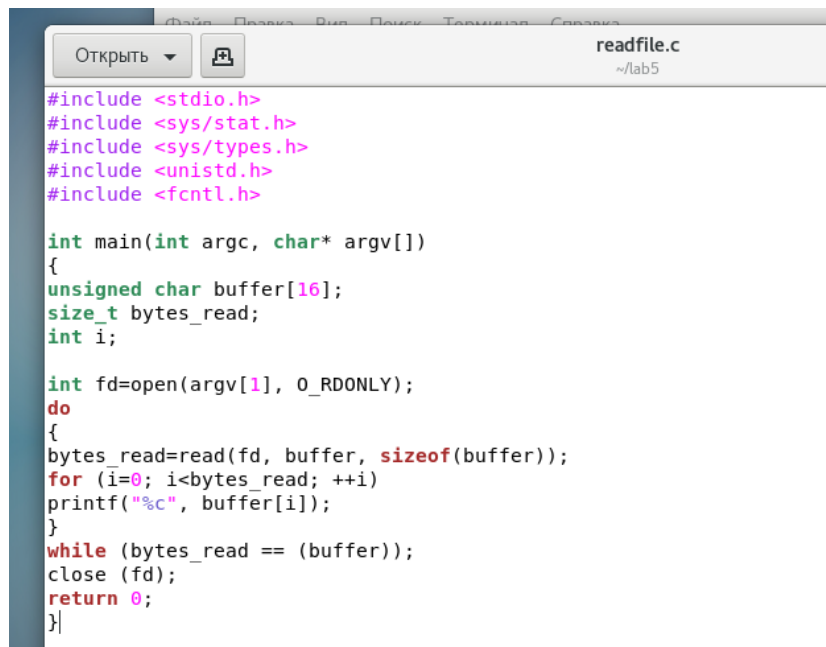
## 12. Проделали тоже самое относительно SetGID-бита.



```
[guest2@iasolodovnikov lab5]$ gedit simpleid2.c
[guest2@iasolodovnikov lab5]$
[guest2@iasolodovnikov lab5]$ gcc simpleid2.c
[guest2@iasolodovnikov lab5]$ gcc simpleid2.c -o simpleid2
[guest2@iasolodovnikov lab5]$ ./simpleid2
e_uid=1002, e_gid=1002
real_uid=1002, real_gid=1002
[guest2@iasolodovnikov lab5]$ su
Пароль:
[root@iasolodovnikov lab5]# chown root:guest simpleid2
[root@iasolodovnikov lab5]# chmod u+s simpleid2
[root@iasolodovnikov lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@iasolodovnikov lab5]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@iasolodovnikov lab5]# chmod g+s simpleid2
[root@iasolodovnikov lab5]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@iasolodovnikov lab5]# exit
exit
[guest2@iasolodovnikov lab5]$
```

Figure 2.5: результат программы simpleid2

## 13. Написали программу readfile.c



```
readfile.c
~/lab5

#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.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 == (buffer));
    close (fd);
    return 0;
}
```

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
```

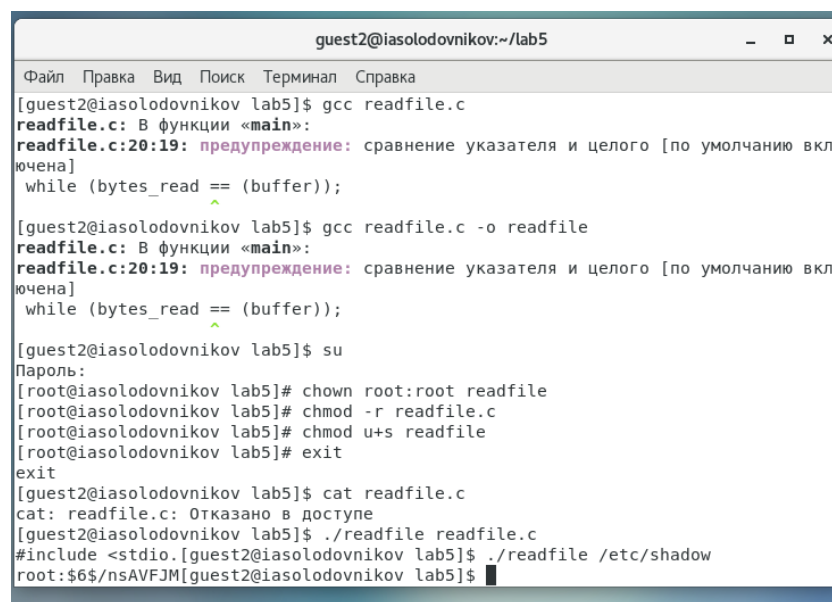
```
chmod 700 /home/guest/readfile.c
```

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



```
guest2@iasolodovnikov:~/lab5
Файл Правка Вид Поиск Терминал Справка
[guest2@iasolodovnikov lab5]$ gcc readfile.c
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого [по умолчанию включена]
    while (bytes_read == (buffer));
                      ^
[guest2@iasolodovnikov lab5]$ gcc readfile.c -o readfile
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого [по умолчанию включена]
    while (bytes_read == (buffer));
                      ^
[guest2@iasolodovnikov lab5]$ su
Пароль:
[root@iasolodovnikov lab5]# chown root:root readfile
[root@iasolodovnikov lab5]# chmod -r readfile.c
[root@iasolodovnikov lab5]# chmod u+s readfile
[root@iasolodovnikov lab5]# exit
exit
[guest2@iasolodovnikov lab5]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest2@iasolodovnikov lab5]$ ./readfile readfile.c
#include <stdio.h>[guest2@iasolodovnikov lab5]$ ./readfile /etc/shadow
root:$6$nsAVFJM[guest2@iasolodovnikov lab5]$
```

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:

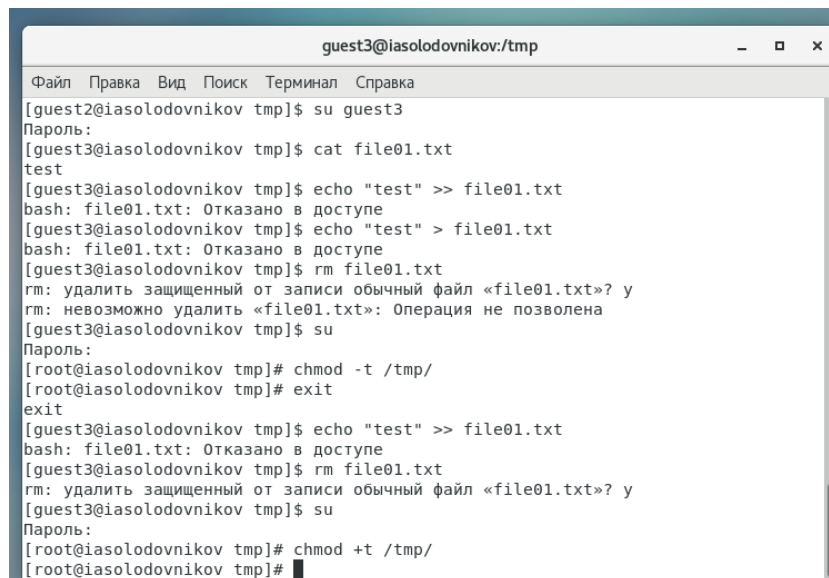
```
ls -l / | grep tmp
```

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
```



```
guest3@iasolodovnikov:/tmp
Файл  Правка  Вид  Поиск  Терминал  Справка
[guest2@iasolodovnikov tmp]$ su guest3
Пароль:
[guest3@iasolodovnikov tmp]$ cat file01.txt
test
[guest3@iasolodovnikov tmp]$ echo "test" >> file01.txt
bash: file01.txt: Отказано в доступе
[guest3@iasolodovnikov tmp]$ echo "test" > file01.txt
bash: file01.txt: Отказано в доступе
[guest3@iasolodovnikov tmp]$ rm file01.txt
rm: удалить защищенный от записи обычный файл «file01.txt»? y
rm: невозможно удалить «file01.txt»: Операция не позволена
[guest3@iasolodovnikov tmp]$ su
Пароль:
[root@iasolodovnikov tmp]# chmod -t /tmp/
[root@iasolodovnikov tmp]# exit
exit
[guest3@iasolodovnikov tmp]$ echo "test" >> file01.txt
bash: file01.txt: Отказано в доступе
[guest3@iasolodovnikov tmp]$ rm file01.txt
rm: удалить защищенный от записи обычный файл «file01.txt»? y
[guest3@iasolodovnikov tmp]$ su
Пароль:
[root@iasolodovnikov tmp]# chmod +t /tmp/
[root@iasolodovnikov 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