

Отчёт по лабораторной работе №5

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

Иванов Владислав НБИ-01-19

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
2.1	Подготовка	5
2.2	Изучение механики SetUID	6
2.3	Исследование Sticky-бита	10
3	Выводы	13
	Список литературы	14

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

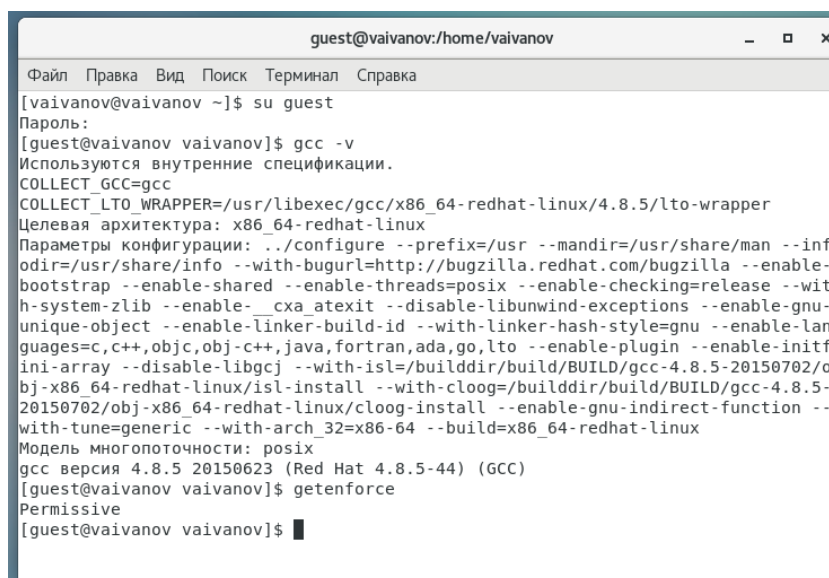
1 Цель работы

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`:

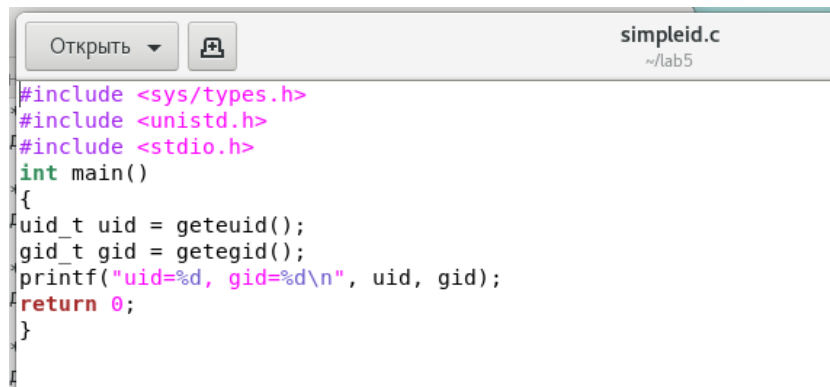


```
guest@vaivanov:/home/vaivanov
Файл Правка Вид Поиск Терминал Справка
[vaivanov@vaivanov ~]$ su guest
Пароль:
[guest@vaivanov vaivanov]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-wrapper
Целевая архитектура: x86_64-redhat-linux
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)
[guest@vaivanov vaivanov]$ getenforce
Permissive
[guest@vaivanov vaivanov]$
```

Figure 2.1: подготовка к работе

2.2 Изучение механики SetUID

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

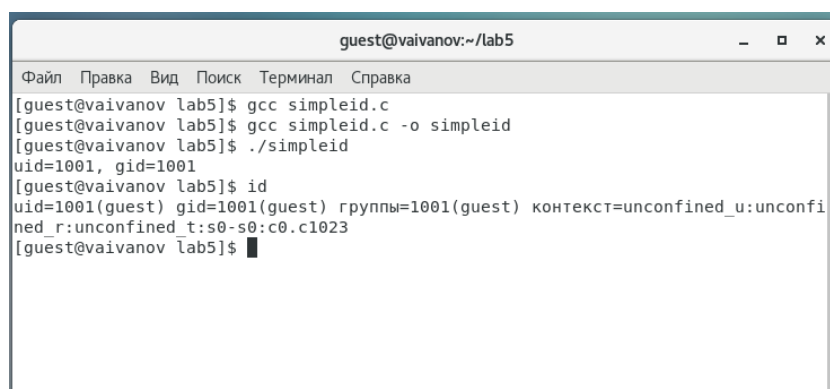


```
simpleid.c
~/lab5

#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
    uid_t uid = getuid();
    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

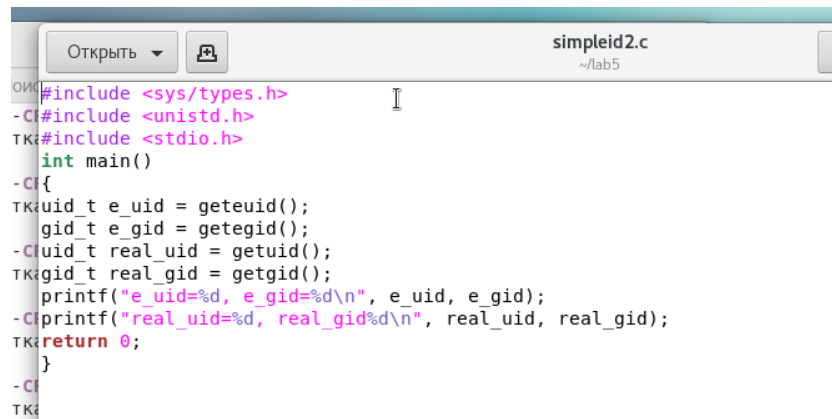


```
guest@vaivanov:~/lab5

Файл  Правка  Вид  Поиск  Терминал  Справка
[guest@vaivanov lab5]$ gcc simpleid.c
[guest@vaivanov lab5]$ gcc simpleid.c -o simpleid
[guest@vaivanov lab5]$ ./simpleid
uid=1001, gid=1001
[guest@vaivanov lab5]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[guest@vaivanov lab5]$
```

Figure 2.3: результат программы simpleid

6. Усложнили программу, добавив вывод действительных идентификаторов.



```
simpleid2.c
~/lab5

#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:

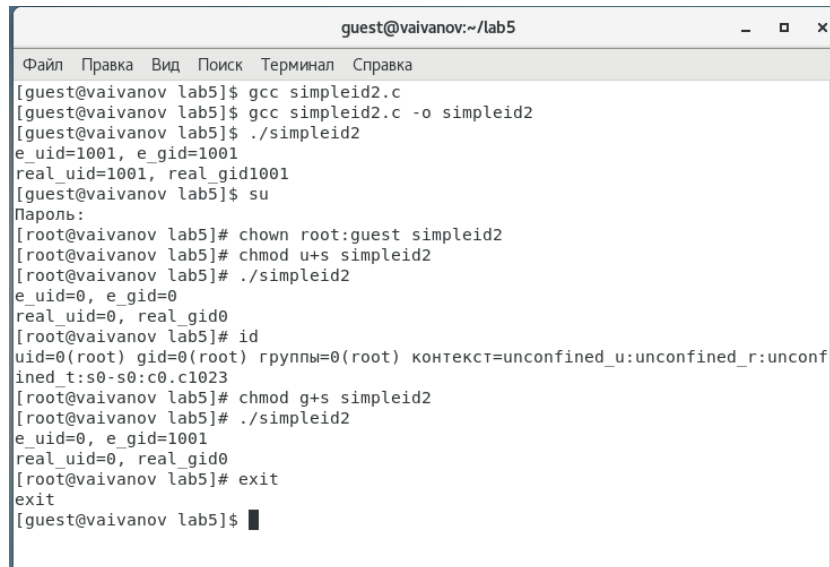
```
ls -l simpleid2
```

11. Запустили simpleid2 и id:

```
./simpleid2
id
```

Результат выполнения программ теперь немного отличается

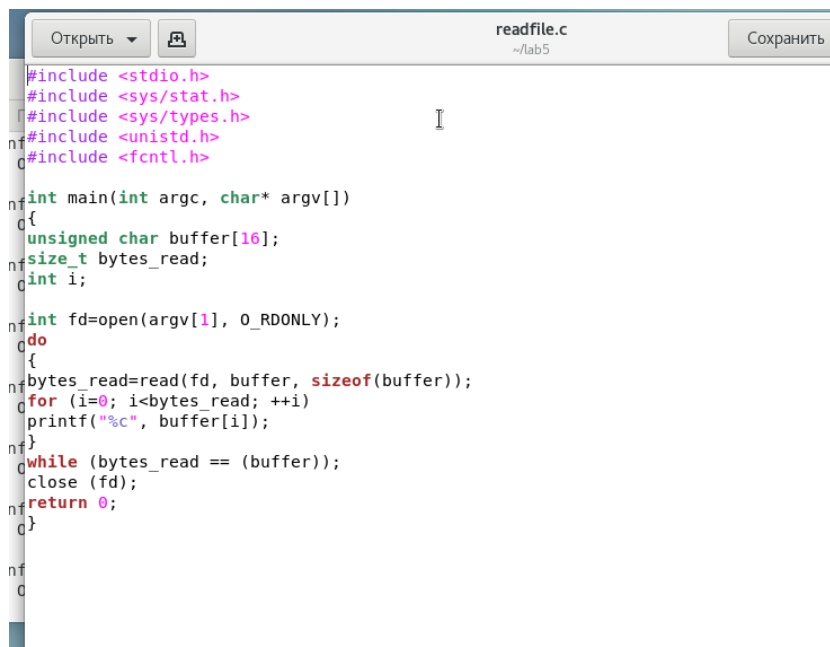
12. Проделали тоже самое относительно SetGID-бита.



```
guest@vaivanov: ~/lab5
Файл Правка Вид Поиск Терминал Справка
[guest@vaivanov lab5]$ gcc simpleid2.c
[guest@vaivanov lab5]$ gcc simpleid2.c -o simpleid2
[guest@vaivanov lab5]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@vaivanov lab5]$ su
Пароль:
[root@vaivanov lab5]# chown root:guest simpleid2
[root@vaivanov lab5]# chmod u+s simpleid2
[root@vaivanov lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@vaivanov lab5]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@vaivanov lab5]# chmod g+s simpleid2
[root@vaivanov lab5]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@vaivanov lab5]# exit
exit
[guest@vaivanov lab5]$
```

Figure 2.5: результат программы simpleid2

13. Написали программу readfile.c



```
Открыть  readfile.c  Сохранить
~/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
```

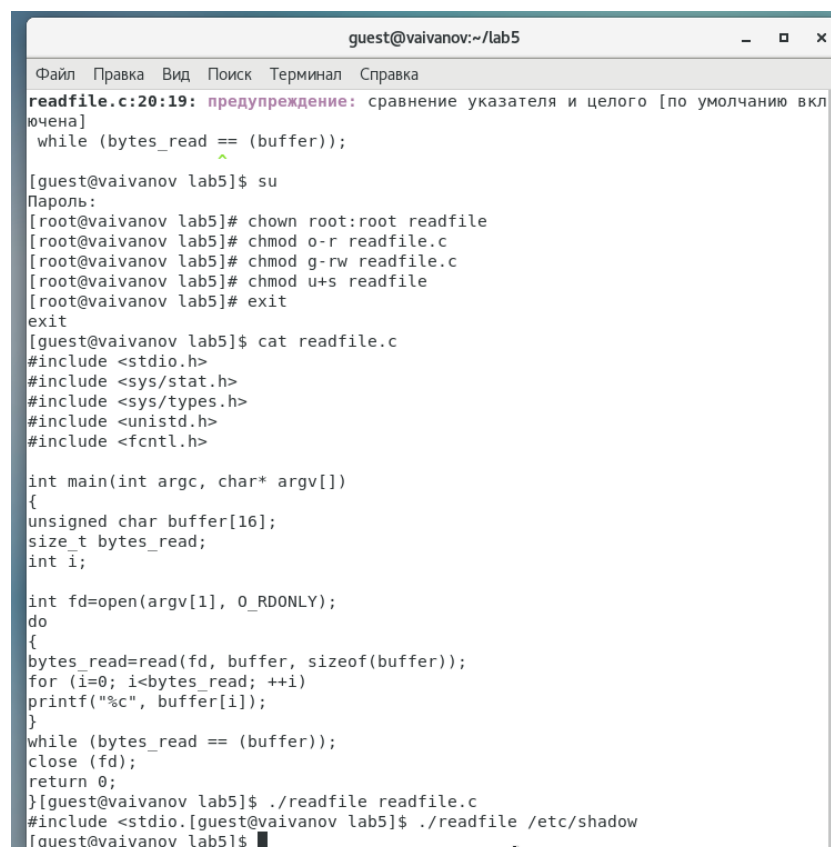
```
chmod 700 /home/guest/readfile.c
```

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



```
guest@vaivanov:~/lab5
Файл  Правка  Вид  Поиск  Терминал  Справка
readfile.c:20:19: предупреждение: сравнение указателя и целого [по умолчанию вкл
ючена]
while (bytes_read == (buffer));
                    ^
[guest@vaivanov lab5]$ su
Пароль:
[root@vaivanov lab5]# chown root:root readfile
[root@vaivanov lab5]# chmod o-r readfile.c
[root@vaivanov lab5]# chmod g-rw readfile.c
[root@vaivanov lab5]# chmod u+s readfile
[root@vaivanov lab5]# exit
exit
[guest@vaivanov lab5]$ cat readfile.c
#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;
}[guest@vaivanov lab5]$ ./readfile readfile.c
#include <stdio.h>[guest@vaivanov lab5]$ ./readfile /etc/shadow
[guest@vaivanov 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
```


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