

Отчёт по лабораторной работе №5 по предмету Информационная безопасность

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

Саттарова Вита Викторовна

Содержание

1	Цели и задачи работы	5
2	Объект и предмет исследования	6
3	Условные обозначения и термины	7
4	Задание	8
4.1	Создание программы	8
4.2	Исследование Sticky-бита	11
5	Теоретическое введение	14
5.1	Подготовка лабораторного стенда	14
5.2	Компилирование программ	15
6	Техническое оснащение и выбранные методы проведения работы	18
7	Выполнение лабораторной работы и полученные результаты	19
8	Анализ результатов	34
9	Заключение и выводы	35
10	Список литературы	36

Список иллюстраций

7.1	Лабораторная работа 5	20
7.2	Задания 1-2 Часть 1	21
7.3	Задания 3-5 Часть 1	22
7.4	Программа 1 Часть 1	22
7.5	Программа 2 Часть 1	23
7.6	Задания 6-7 Часть 1	24
7.7	Задания 8-9 Часть 1	24
7.8	Задания 10-11 Часть 1	25
7.9	Задание 12 Часть 1	26
7.10	Задание 13 Часть 1	27
7.11	Задания 14-18 Часть 1	28
7.12	Здание 19 Часть 1	29
7.13	Задания 1-3 Часть 2	30
7.14	Задания 4-9 Часть 2	31
7.15	Задания 10-14 Часть 2	32
7.16	Задание 15 Часть 2	33

Список таблиц

1 Цели и задачи работы

Цели:

- Закрепить теоретические основы дискреционного разграничения доступа в современных системах с открытым кодом на базе ОС Linux.
- Изучить механизмы изменения идентификаторов, применения SetUID- и Sticky-битов.
- Получить практические навыки работы в консоли с дополнительными атрибутами.
- Рассмотреть работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Задачи:

- Выполнить все пункты, указанные в методических рекомендациях к лабораторной работе.
- Ответить на вопросы, заданные в методических рекомендациях к лабораторной работе.
- Выполняя задания, изучить особенности SetUID-.
- Выполняя задания, изучить особенности Sticky-битов.
- Написать отчёт, проанализировав результаты, полученные в ходе выполнения лабораторной работы.

2 Объект и предмет исследования

Объект исследования: использование SetUID- и Sticky-битов в ОС Linux для обеспечения безопасности.

Предмет исследования: идентификаторы, SetUID-, Sticky-бит.

3 Условные обозначения и термины

Условные обозначения

- ОС - операционная система

Термины

- Дискреционное разграничение доступа
- идентификаторы
- SetUID-
- Sticky-бит

4 Задание

4.1 Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте 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;
}
```

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.

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;
}
```

Получившуюся программу назовите simpleid2.c.

7. Скомпилируйте и запустите simpleid2.c:

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

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.

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.

13. Создайте программу `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;
}

```

14. Откомпилируйте её.

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

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.

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.

4.2 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
```

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?

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?

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?

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
```

13. Повторите предыдущие шаги. Какие наблюдаются изменения?

14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Ваши наблюдения занесите в отчёт.

15. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp:

```
su -
```

```
chmod +t /tmp
```

```
exit
```

Более подробно о работе см. в [1].

5 Теоретическое введение

5.1 Подготовка лабораторного стенда

Помимо прав администратора для выполнения части заданий потребуются средства разработки приложений. В частности, при подготовке стенда следует убедиться, что в системе установлен компилятор gcc (для этого, например, можно ввести команду `gcc -v`). Если же gcc не установлен, то его необходимо установить, например, командой `yum install gcc`, которая определит зависимости и установит следующие пакеты: `gcc`, `cloogpp`, `cpp`, `glibc-devel`, `glibc-headers`, `kernel-headers`, `libgomp`, `ppl`, `cloog-ppl`, `gcc`, `glibc-devel`, `glibc-headers`, `kernel-headers`, `libgomp`, `libstdc++-devel`, `mpfr`, `ppl`, `glibc`, `glibc-common`, `libgcc`, `libstdc++`.

Файловая система, где располагаются домашние директории и файлы пользователей (в частности, пользователя `guest`), не должна быть смонтирована с опцией `nosuid`.

Так как программы с установленным битом SetUID могут представлять большую брешь в системе безопасности, в современных системах используются дополнительные механизмы защиты. Проследите, чтобы система защиты SELinux не мешала выполнению заданий работы. Если вы не знаете, что это такое, просто отключите систему запретов до очередной перезагрузки системы командой `setenforce 0`. После этого команда `getenforce` должна выводить `Permissive`.

5.2 Компилирование программ

Для выполнения четвёртой части задания вам потребуются навыки программирования, а именно, умение компилировать простые программы, написанные на языке C (C++), используя интерфейс CLI.

Само по себе создание программ не относится к теме, по которой выполняется работа, а является вспомогательной частью, позволяющей увидеть, как реализуются на практике те или иные механизмы дискреционного разграничения доступа. Если при написании (или исправлении существующих) скриптов на `bash`-е у большинства системных администраторов не возникает проблем, то процесс компилирования, как показывает практика, вызывает необоснованные затруднения.

Компиляторы, доступные в Linux-системах, являются частью коллекции GNU-компиляторов, известной как GCC (GNU Compiler Collection, подробнее см. <http://gcc.gnu.org>). В неё входят компиляторы языков C, C++, Java, Objective-C, Fortran и Chill. Будем использовать лишь первые два.

Компилятор языка C называется `gcc`. Компилятор языка C++ называется `g++` и запускается с параметрами почти так же, как `gcc`. Проверить это можно следующими командами: `whereis gcc`, `whereis g++`

Первый шаг заключается в превращении исходных файлов в объектный код:

```
gcc -c file.c
```

В случае успешного выполнения команды (отсутствие ошибок в коде) полученный объектный файл будет называться `file.o`.

Объектные файлы невозможно запускать и использовать, поэтому после компиляции для получения готовой программы объектные файлы необходимо компоновать. Компоновать можно один или несколько файлов. В случае использования хотя бы одного из файлов, написанных на C++, компоновка производится с помощью компилятора `g++`. Строго говоря, это тоже не вполне верно. Компоновка объектного кода, сгенерированного чем бы то ни было (хоть вручную), производится линкером `ld`, `g++` его просто вызывает изнутри. Если же все файлы

написаны на языке C, нужно использовать компилятор gcc.

Например, так: `gcc -o program file.o` В случае успешного выполнения команды будет создана программа `program` (исполняемый файл формата ELF с установленным атрибутом `+x`).

Компилирование — это процесс. Компилятор gcc (g++) имеет множество параметров, влияющих на процесс компиляции. Он поддерживает различные режимы оптимизации, выбор платформы назначения и пр. Также возможно использование make-файлов (Makefile) с помощью утилиты make для упрощения процесса компиляции.

Такое решение подойдёт лишь для простых случаев. Если говорить про пример выше, то компилирование одного файла из двух шагов можно сократить вообще до одного, например: `gcc file.c`

В этом случае готовая программа будет иметь название `a.out`. Механизм компилирования программ в данной работе не мог быть не рассмотрен потому, что использование программ, написанных на bash, для изучения SetUID- и SetGID- битов, не представляется возможным. Связано это с тем, что любая bash-программа интерпретируется в процессе своего выполнения, т.е. существует сторонняя программа-интерпретатор, которая выполняет считывание файла сценария и выполняет его последовательно. Сам интерпретатор выполняется с правами пользователя, его запустившего, а значит, и выполняемая программа использует эти права. При этом интерпретатору абсолютно всё равно, установлены SetUID-, SetGID-биты у текстового файла сценария, атрибут разрешения запуска «x» или нет. Важно, чтобы был установлен лишь атрибут, разрешающий чтение «r».

Также не важно, был ли вызван интерпретатор из командной строки (запуск файла, как `bash file1.sh`), либо внутри файла была указана строчка `#!/bin/bash`.

Логично спросить: если установление SetUID- и SetGID- битов на сценарий не приводит к нужному результату как с исполняемыми файлами, то что мешает установить эти биты на сам интерпретатор? Ничего не мешает, только их уста-

новление приведёт к тому, что, так как владельцем `/bin/bash` является `root`: `ls -l /bin/bash` все сценарии, выполняемые с использованием `/bin/bash`, будут иметь возможности суперпользователя — совсем не тот результат, который хотелось бы видеть.

Если сомневаетесь в выше сказанном, создайте простой файл `progl.sh` следующего содержания:

```
#!/bin/bash
/usr/bin/id /usr/bin/whoami
```

и попробуйте поменять его атрибуты в различных конфигурациях. Подход вида: сделать копию `/bin/bash`, для нее `chown user:users` и потом SUID также плох, потому что это позволит запускать любые команды от пользователя `user`.

Более подробно о работе см. в [1].

6 Техническое оснащение и выбранные методы проведения работы

Техническое оснащение

- Ноутбук
- CentOS
- Интернет

Методы проведения работы

- Изучение методической информации
- Выполнение заданий в соответствии с указаниями
- Анализ результатов
- Ответы на вопросы, заданные в задании
- Обобщение проведённой деятельности

7 Выполнение лабораторной работы и полученные результаты

1. Скачала и ознакомилась с методическими указаниями к лабораторной работе (рис. 7.1).

Лабораторная работа № 5. Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

5.1. 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.¹

5.2. Указания к работе

5.2.1. Подготовка лабораторного стенда

Помимо прав администратора для выполнения части заданий потребуются средства разработки приложений. В частности, при подготовке стенда следует убедиться, что в системе установлен компилятор gcc (для этого, например, можно ввести команду gcc -v). Если же gcc не установлен, то его необходимо установить, например, командой

```
yum install gcc
```

Рис. 7.1: Лабораторная работа 5

2. Выполнила следующие задания:

- Войдите в систему от имени пользователя guest.
- Создайте 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;
}
```

(рис. 7.2).

```
[vvsattarova@vvsattarova ~]$ su guest
Пароль:
[guest@vvsattarova vvsattarova]$ touch simpleid.c
touch: невозможно выполнить touch для «simpleid.c»: Отказано в доступе
[guest@vvsattarova vvsattarova]$ cd /home/guest/dir1
[guest@vvsattarova dir1]$ ls
file1
[guest@vvsattarova dir1]$ touch simpleid.c
[guest@vvsattarova dir1]$ gedit simpleid.c
```

Рис. 7.2: Задания 1-2 Часть 1

3. Выполнила следующие задания:

- Скомпилируйте программу и убедитесь, что файл программы создан: `gcc simpleid.c -o simpleid`
- Выполните программу simpleid: `./simpleid`
- Выполните системную программу id: `id`

и сравните полученный вами результат с данными предыдущего пункта задания - вывод одинаковый (рис. 7.3).

```
[guest@vvsattarova dir1]$ gcc simpleid.c -o simpleid
[guest@vvsattarova dir1]$ ./simpleid
uid=1001, gid=1001
[guest@vvsattarova dir1]$ ls
file1  simpleid  simpleid.c
[guest@vvsattarova dir1]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[guest@vvsattarova dir1]$ █
```

Рис. 7.3: Задания 3-5 Часть 1

4. Код первой программы (рис. 7.4).

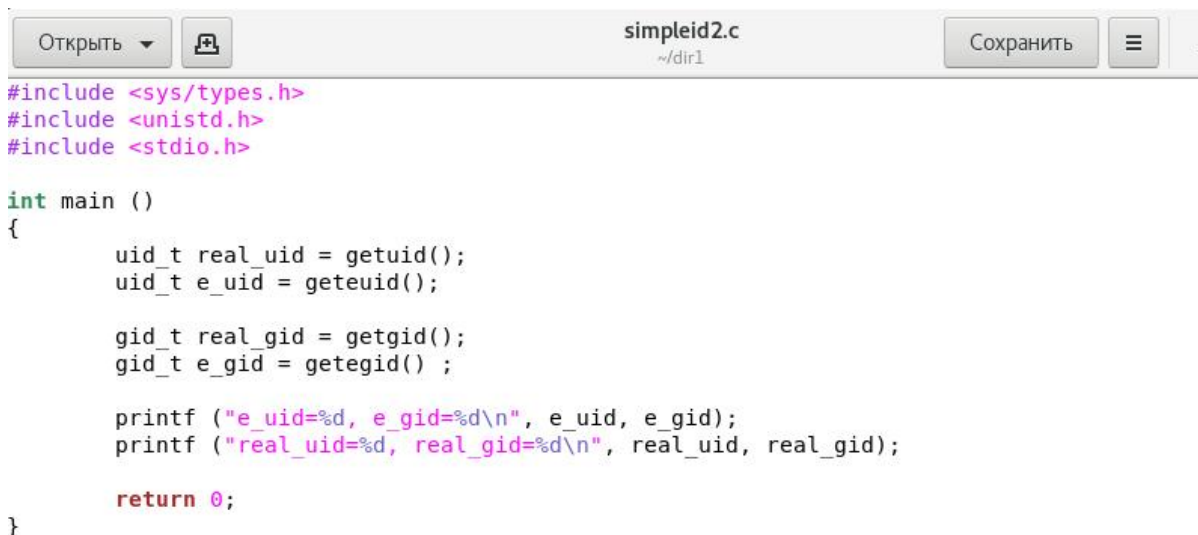


```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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

Рис. 7.4: Программа 1 Часть 1

5. Код второй программы (рис. 7.5).



```
#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. 7.5: Программа 2 Часть 1

6. Выполнила следующие задания:

- 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;
}
```

}

Получившуюся программу назовите simpleid2.c.

- Скомпилируйте и запустите simpleid2.c:

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

(рис. 7.6).

```
[guest@vvsattarova dir1]$ gcc simpleid2.c -o simpleid2
[guest@vvsattarova dir1]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@vvsattarova dir1]$
```

Рис. 7.6: Задания 6-7 Часть 1

7. Выполнила следующие задания:

- От имени суперпользователя выполните команды:

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

- Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды - первая меняет владельца файла, вторая меняет SetUID-бит. (рис. 7.7).

```
[root@vvsattarova ~]# chown root:guest /home/guest/dir1/simpleid2
[root@vvsattarova ~]# chmod u+s /home/guest/dir1/simpleid2
[root@vvsattarova ~]#
```

Рис. 7.7: Задания 8-9 Часть 1

8. Выполнила следующие задания:

- Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2:

```
ls -l simpleid2
```

- Запустите simpleid2 и id:

```
./simpleid2
```

```
id
```

Сравните результаты - отличается e_uid и UID (рис. 7.8).

```
[guest@vvsattarova dir1]$ ls -l simpleid2
-rwsrwxr-x. 1 root guest 8576 окт  7 17:52 simpleid2
[guest@vvsattarova dir1]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@vvsattarova dir1]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[guest@vvsattarova dir1]$ █
```

Рис. 7.8: Задания 10-11 Часть 1

9. Выполнила следующие задания:

- Прodelайте тоже самое относительно SetGID-бита - результаты не изменились (рис. 7.9).

```
root@vvsattarova:~  
Файл Правка Вид Поиск Терминал Справка  
chmod: невозможно получить доступ к «/home/guest/simpleid2»: Нет такого файла ил  
и каталога  
[root@vvsattarova ~]# chmod g+s /home/guest/dir1/simpleid2  
[root@vvsattarova ~]#  
[guest@vvsattarova dir1]$ ls -l simpleid2  
-rwsrwsr-x. 1 root guest 8576 окт  7 17:52 simpleid2  
[guest@vvsattarova dir1]$ ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@vvsattarova dir1]$ id  
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi  
ned_r:unconfined_t:s0-s0:c0.c1023  
[guest@vvsattarova dir1]$
```

Рис. 7.9: Задание 12 Часть 1

10. Выполнила следующие задания:

- Создайте программу 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;
}

```

(рис. 7.10).

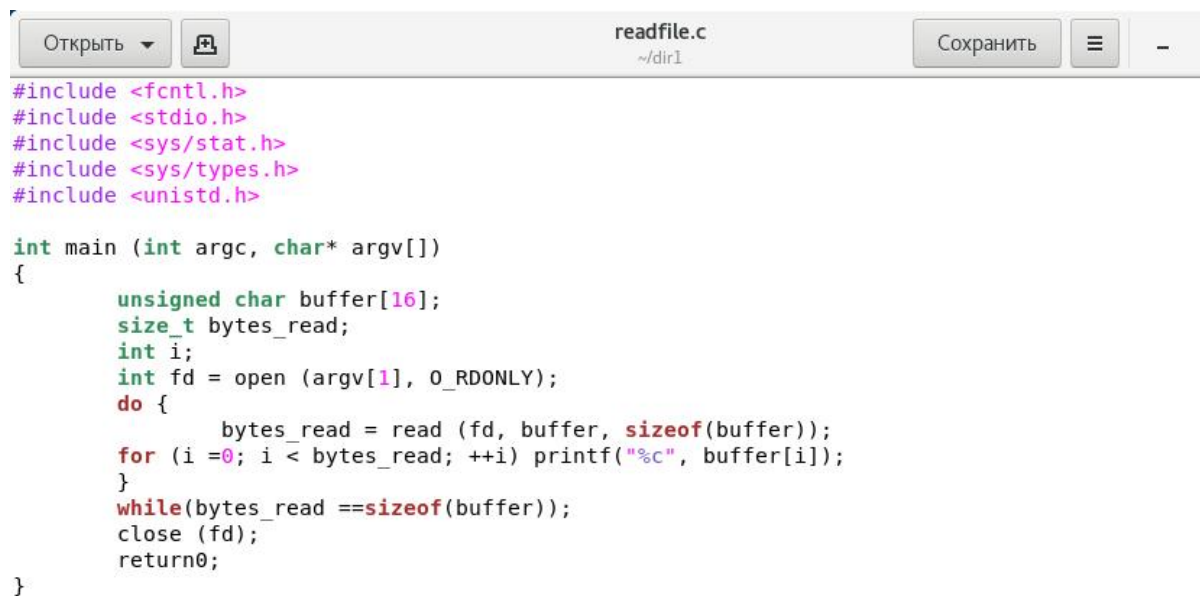


Рис. 7.10: Задание 13 Часть 1

11. Выполнила следующие задания:

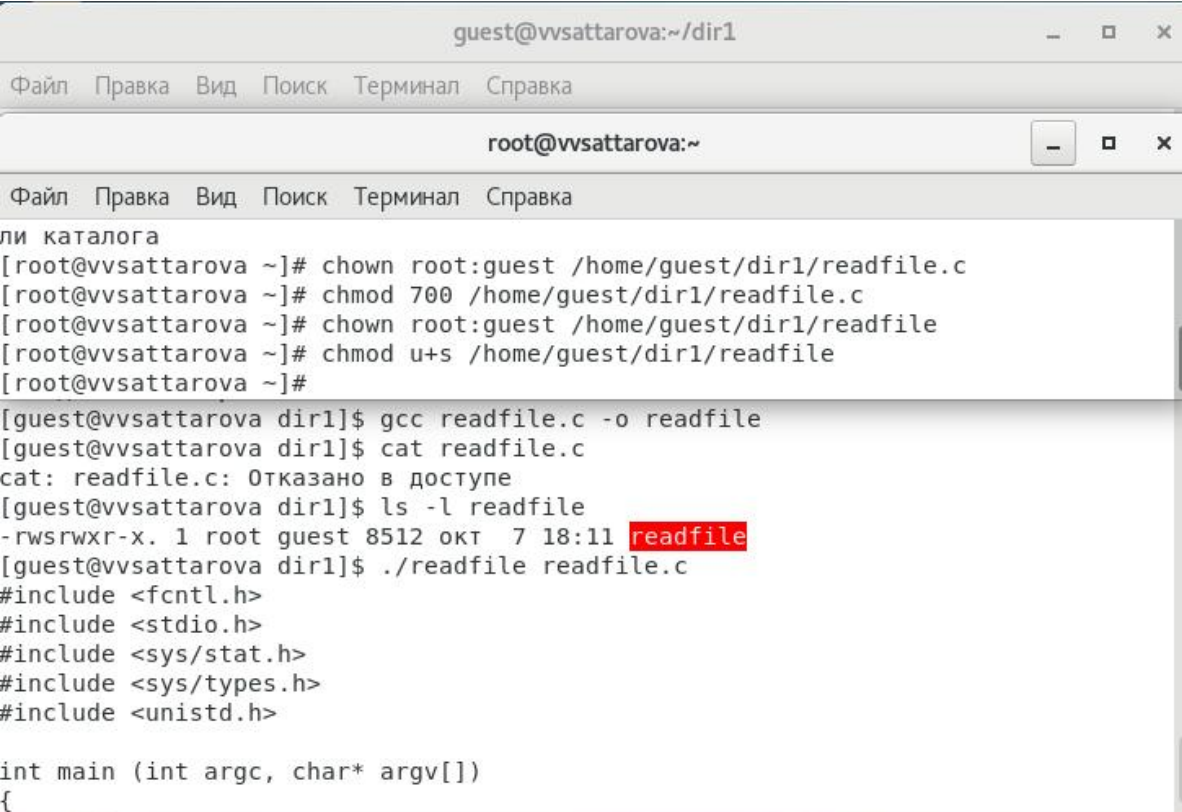
- Откомпилируйте её.

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

- 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.

- Проверьте, что пользователь guest не может прочитать файл readfile.c.
- Смените у программы readfile владельца и установите SetU'D-бит.
- Проверьте, может ли программа readfile прочитать файл readfile.c? Да, может.

(рис. 7.11).



```

guest@vvsattarova:~/dir1
Файл  Правка  Вид  Поиск  Терминал  Справка

root@vvsattarova:~
Файл  Правка  Вид  Поиск  Терминал  Справка

ли каталога
[root@vvsattarova ~]# chown root:guest /home/guest/dir1/readfile.c
[root@vvsattarova ~]# chmod 700 /home/guest/dir1/readfile.c
[root@vvsattarova ~]# chown root:guest /home/guest/dir1/readfile
[root@vvsattarova ~]# chmod u+s /home/guest/dir1/readfile
[root@vvsattarova ~]#
[guest@vvsattarova dir1]$ gcc readfile.c -o readfile
[guest@vvsattarova dir1]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@vvsattarova dir1]$ ls -l readfile
-rwsrwxr-x. 1 root guest 8512 окт  7 18:11 readfile
[guest@vvsattarova dir1]$ ./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[])
{

```

Рис. 7.11: Задания 14-18 Часть 1

12. Выполнила следующие задания:

- Проверьте, может ли программа readfile прочитать файл /etc/shadow? Да, может (рис. 7.12).

```
[guest@vvsattarova dir1]$ ./readfile /etc/shadow
root:$6$ms.v5Y6Q0wkXgaRa$I.Yl2r9a3a2CjWQiSYi.mxIXHXB9JYw9CTgfHgG/XlZH6fzMf
C1cY6YaqyfZe8jajNQ/DCvjwpQWo/o0::0:99999:7:::
bin:*:18353:0:99999:7:::
daemon:*:18353:0:99999:7:::
adm:*:18353:0:99999:7:::
lp:*:18353:0:99999:7:::
sync:*:18353:0:99999:7:::
shutdown:*:18353:0:99999:7:::
halt:*.18353:0:99999:7:::
```

Рис. 7.12: Здание 19 Часть 1

13. Выполните следующие задания:

- Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду

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

- От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test:

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

- 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
```

(рис. 7.13).

```
[guest@vvsattarova dir1]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 окт 7 18:17 tmp
[guest@vvsattarova dir1]$ echo "test" > /tmp/file01.txt
bash: echo: команда не найдена...
[guest@vvsattarova dir1]$ echo "test" > /tmp/file01.txt
[guest@vvsattarova dir1]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 окт 7 18:17 /tmp/file01.txt
[guest@vvsattarova dir1]$ chmod o+rw /tmp/file01.txt
[guest@vvsattarova dir1]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 окт 7 18:17 /tmp/file01.txt
[guest@vvsattarova dir1]$ █
```

Рис. 7.13: Задания 1-3 Часть 2

14. Выполнила следующие задания:

- От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt:

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

- От пользователя guest2 попробуйте дозаписать в файл

/tmp/file01.txt слово test2 командой

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

Удалось ли вам выполнить операцию? Да.

- Проверьте содержимое файла командой

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

- 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? Да.

- Проверьте содержимое файла командой

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

- От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой

```
rm /tmp/file01.txt
```

Удалось ли вам удалить файл? Нет.

(рис. 7.14).

```
[vvsattarova@vvsattarova ~]$ su guest2
Пароль:
[guest2@vvsattarova vvsattarova]$ cat /tmp/file01.txt
test
[guest2@vvsattarova vvsattarova]$ echo "test2" > /tmp/file01.txt
[guest2@vvsattarova vvsattarova]$ cat /tmp/file01.txt
test2
[guest2@vvsattarova vvsattarova]$ echo "test2" >> /tmp/file01.txt
[guest2@vvsattarova vvsattarova]$ cat /tmp/file01.txt
test2
test2
[guest2@vvsattarova vvsattarova]$ echo "test3" > /tmp/file01.txt
[guest2@vvsattarova vvsattarova]$ cat /tmp/file01.txt
test3
[guest2@vvsattarova vvsattarova]$ rm /tmp/file01.txt
rm: невозможно удалить «/tmp/file01.txt»: Нет такого файла или каталога
[guest2@vvsattarova vvsattarova]$ rm /tmp/file01.txt
rm: невозможно удалить «/tmp/file01.txt»: Нет такого файла или каталога
[guest2@vvsattarova vvsattarova]$ cat /tmp/file01.txt
test3
[guest2@vvsattarova vvsattarova]$ rm /tmp/file01.txt
rm: невозможно удалить «/tmp/file01.txt»: Операция не позволена
[guest2@vvsattarova vvsattarova]$ █
```

Рис. 7.14: Задания 4-9 Часть 2

15. Выполнила следующие задания:

- Повысьте свои права до суперпользователя следующей командой

```
su -
```

и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp:

```
chmod -t /tmp
```


- Покиньте режим суперпользователя командой

`exit`

- От пользователя `guest2` проверьте, что атрибута `t` у директории `/tmp` нет:

`ls -l / | grep tmp`

- Повторите предыдущие шаги. Какие наблюдаются изменения? Изменений практически нет, все действия выполняются, включая удаление.
- Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Да, удалось.

(рис. 7.15).

```
[guest2@vvsattarova vvsattarova]$ su -
Пароль:
Последний вход в систему:Сб окт  7 17:54:28 GMT 2023на pts/0
[root@vvsattarova ~]# chmod -t /tmp
[root@vvsattarova ~]# exit
logout
[guest2@vvsattarova vvsattarova]$ ls -l / | grep tmp
drwxrwxrwx. 17 root root 4096 окт  7 18:26 tmp
[guest2@vvsattarova vvsattarova]$ echo "test2" >> /tmp/file01.txt
[guest2@vvsattarova vvsattarova]$ echo "test2" >> /tmp/file01.txt
[guest2@vvsattarova vvsattarova]$ cat /tmp/file01.txt
test3
test2
test2
[guest2@vvsattarova vvsattarova]$ echo "test3" > /tmp/file01.txt
[guest2@vvsattarova vvsattarova]$ cat /tmp/file01.txt
test3
[guest2@vvsattarova vvsattarova]$ rm /tmp/file01.txt
[guest2@vvsattarova vvsattarova]$ ls -l / | grep tmp
drwxrwxrwx. 17 root root 4096 окт  7 18:27 tmp
[guest2@vvsattarova vvsattarova]$ ls -l / | grep tmp/file01.txt
[guest2@vvsattarova vvsattarova]$
```

Рис. 7.15: Задания 10-14 Часть 2

16. Выполнила следующие задания:

- Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp:

```
su -  
chmod +t /tmp  
exit
```

(рис. 7.16).

```
[guest2@vvsattarova vvsattarova]$ su -  
Пароль:  
Последний вход в систему:Сб окт  7 18:25:36 GMT 2023на pts/2  
[root@vvsattarova ~]# chmod +t /tmp  
[root@vvsattarova ~]# exit  
logout  
[guest2@vvsattarova vvsattarova]$
```

Рис. 7.16: Задание 15 Часть 2

8 Анализ результатов

Таким образом, были выполнены задания по изучению идентификаторов, SetUID-бита, Sticky-бита.

9 Заключение и выводы

Таким образом, в ходе выполнения лабораторной работы было сделано следующее:

- Вспомнены теоретические основы атрибутов файлов и директорий в ОС Linux.
- Изучены механизмы изменения идентификаторов, применения SetUID- и Sticky-битов.
- Получены практические навыки работы в консоли с дополнительными атрибутами.
- Рассмотрена работа механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.
- Написан отчёт к лабораторной работе.

10 Список литературы

[1]

1. Информационная безопасность [Электронный ресурс]. Российский университет дружбы народов, 2023. URL: https://esystem.rudn.ru/pluginfile.php/2090279/mod_resource/content/2/005-lab_discret_sticky.pdf.