

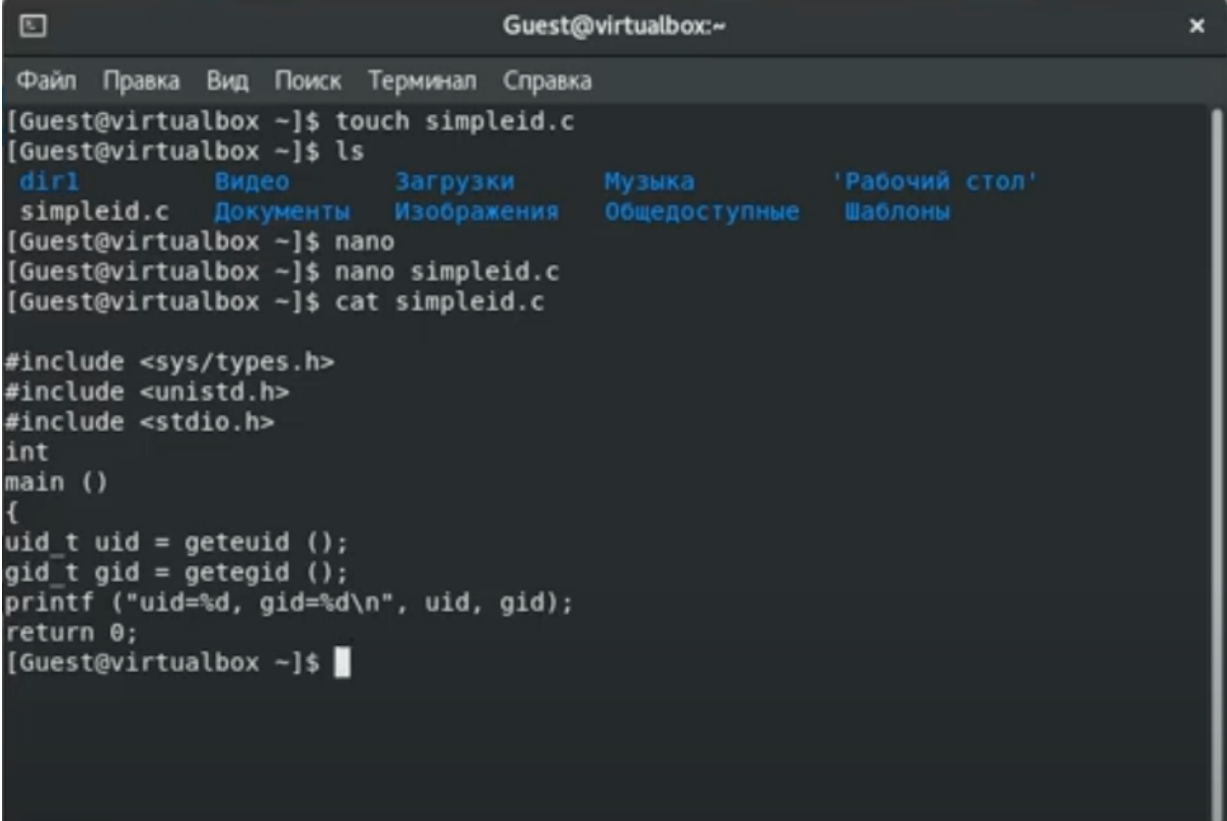
Лабораторная работа 5

Цель работы

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в консоли дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Ход работы

От имени пользователя guest создам файл simpleid.c и создадим программу



```
Guest@virtualbox:~  
Файл Правка Вид Поиск Терминал Справка  
[Guest@virtualbox ~]$ touch simpleid.c  
[Guest@virtualbox ~]$ ls  
dir1 Видео Загрузки Музыка 'Рабочий стол'  
simpleid.c Документы Изображения Общедоступные Шаблоны  
[Guest@virtualbox ~]$ nano  
[Guest@virtualbox ~]$ nano simpleid.c  
[Guest@virtualbox ~]$ cat 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;  
}[Guest@virtualbox ~]$
```

{ #fig:001 width=70% }

Скомпилируем программу:

```
[Guest@virtualbox ~]$ gcc simpleid.c -o simpleid
[Guest@virtualbox ~]$ ls
dirl      simpleid.c  Документы  Изображения  Общедоступные  Шаблоны
simpleid  Видео      Загрузки  Музыка      'Рабочий стол'
```

{ #fig:002 width=70% }

Запускаем simpleid.c и сверяем с результатом команды id

```
[Guest@virtualbox ~]$ ./simpleid
uid=1002, gid=1003
[Guest@virtualbox ~]$ id
uid=1002(Guest) gid=1003(Guest) группы=1003(Guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[Guest@virtualbox ~]$
```

{ #fig:003 width=70% }

Получили одинаковые результаты параметров.

Усложним программу "simpleid.c", добавив вывод действительных идентификаторов

```
[Guest@virtualbox ~]$ nano simpleid2.c
[Guest@virtualbox ~]$ cat simpleid2.c
#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;
}

[Guest@virtualbox ~]$
```

{ #fig:004 width=70% }

Скомпилируем запустим simpleid2.c

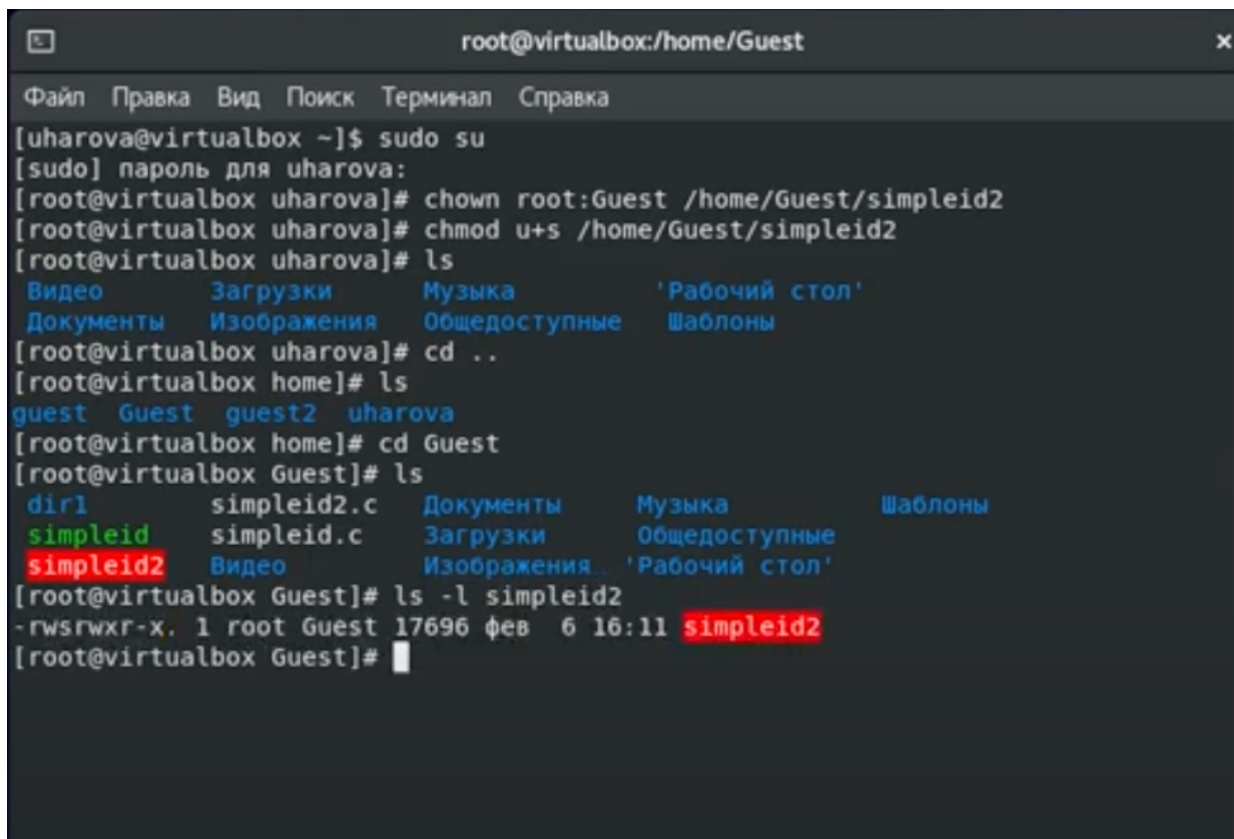
```
[Guest@virtualbox ~]$ gcc simpleid2.c -o simpleid2
[Guest@virtualbox ~]$ ./simpleid2
e_uid=1002, e_gid=1003
real_uid=1002, real_gid=1003
```

{ #fig:005 width=70% }

Теперь стало видно не только текущих группу и пользователя, но и владельца файла.

Сменим владельца файла "simpleid2" и добавим ему атрибут SetUID

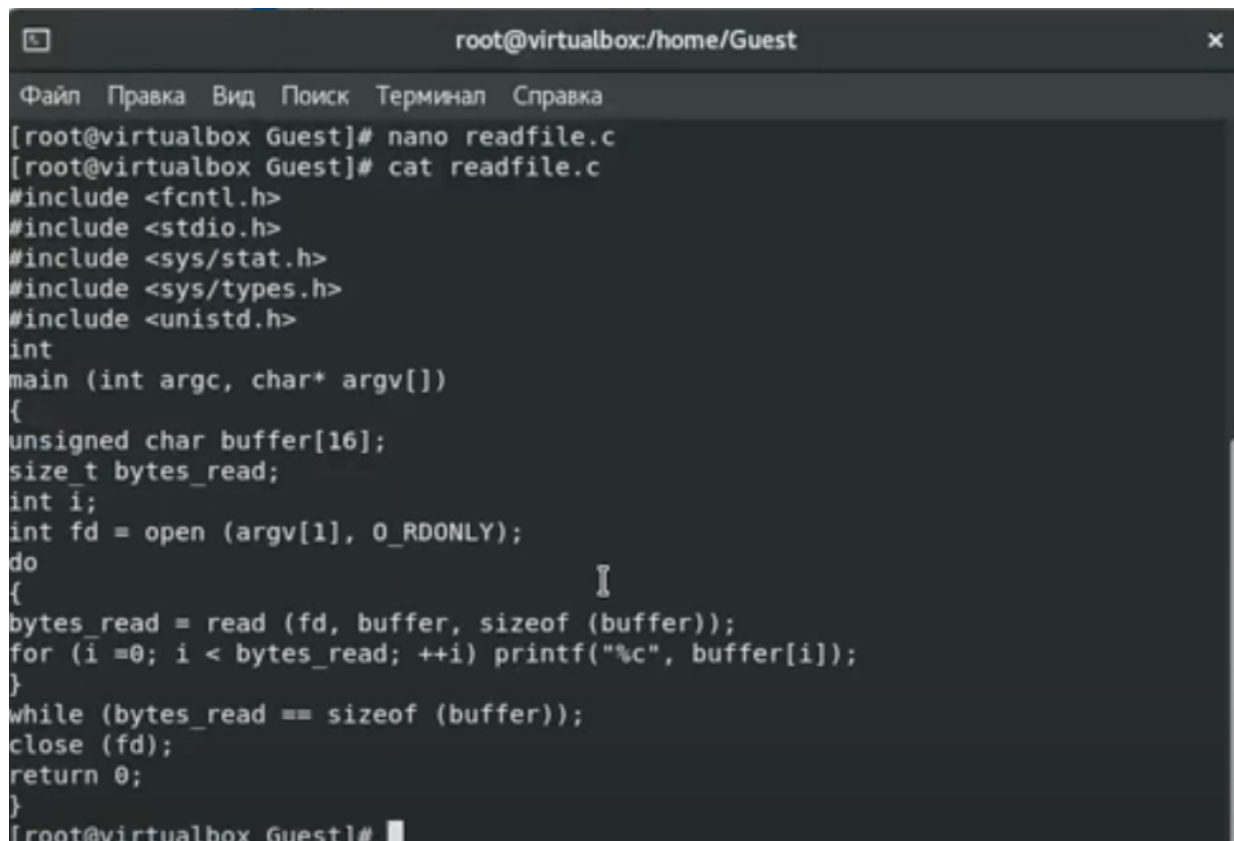
Запустим simpleid2 и id. Получаем идентичные параметры:

A terminal window titled 'root@virtualbox:/home/Guest' with a menu bar (Файл, Правка, Вид, Поиск, Терминал, Справка). The user 'uharova' runs 'sudo su' and then 'chown root:Guest /home/Guest/simpleid2'. Next, they run 'chmod u+s /home/Guest/simpleid2'. They then navigate to the 'Guest' directory and run 'ls -l simpleid2', which shows the file with permissions '-rwsrwxr-x', owner 'root', group 'Guest', size '17696', and timestamp 'фев 6 16:11'. The filename 'simpleid2' is highlighted in red in the original image.

```
root@virtualbox:/home/Guest
[uharova@virtualbox ~]$ sudo su
[sudo] пароль для uharova:
[root@virtualbox uharova]# chown root:Guest /home/Guest/simpleid2
[root@virtualbox uharova]# chmod u+s /home/Guest/simpleid2
[root@virtualbox uharova]# ls
Видео      Загрузки      Музыка      'Рабочий стол'
Документы  Изображения  Общедоступные  Шаблоны
[root@virtualbox uharova]# cd ..
[root@virtualbox home]# ls
guest  Guest  guest2  uharova
[root@virtualbox home]# cd Guest
[root@virtualbox Guest]# ls
dirl      simpleid2.c  Документы  Музыка      Шаблоны
simpleid   simpleid.c   Загрузки   Общедоступные
simpleid2  Видео       Изображения  'Рабочий стол'
[root@virtualbox Guest]# ls -l simpleid2
-rwsrwxr-x. 1 root Guest 17696 фев 6 16:11 simpleid2
[root@virtualbox Guest]#
```

{ #fig:006 width=70% }

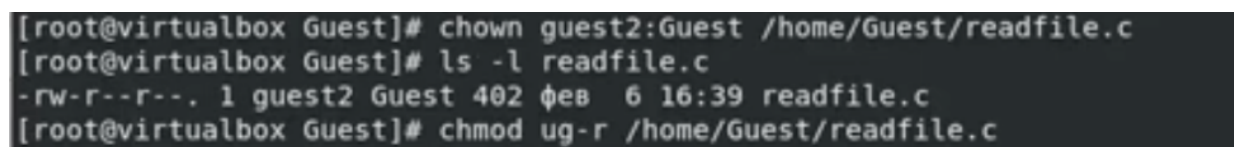
Создадим файл readfile.c



```
root@virtualbox:/home/Guest
Файл  Правка  Вид  Поиск  Терминал  Справка
[root@virtualbox Guest]# nano readfile.c
[root@virtualbox Guest]# cat 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;
}
[root@virtualbox Guest]#
```

{ #fig:007 width=70% }

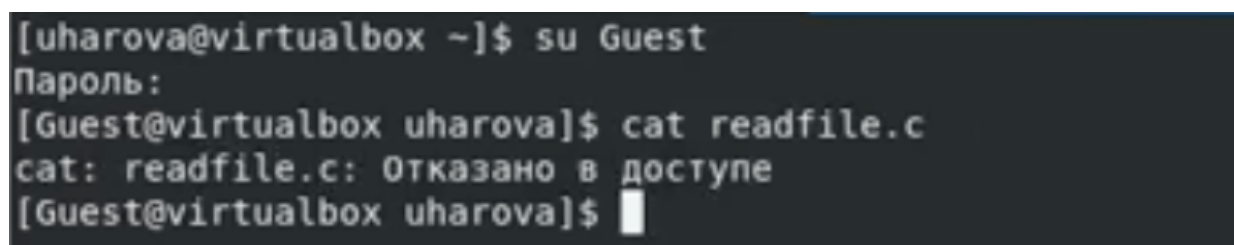
Скомпилируем и изменим права, чтобы только root мог прочитать его, а guest -- нет



```
[root@virtualbox Guest]# chown guest2:Guest /home/Guest/readfile.c
[root@virtualbox Guest]# ls -l readfile.c
-rw-r--r--. 1 guest2 Guest 402 фев  6 16:39 readfile.c
[root@virtualbox Guest]# chmod ug-r /home/Guest/readfile.c
```

{ #fig:008 width=70% }

Проверим, что guest не может прочесть readfile.c



```
[uharova@virtualbox ~]$ su Guest
Пароль:
[Guest@virtualbox uharova]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[Guest@virtualbox uharova]$
```

{ #fig:009 width=70% }

Выясним, что на директории /tmp установлен атрибут Sticky:

```
[root@virtualbox Guest]# ls -l / |grep tmp
drwxrwxrwt. 19 root root 4096 фев  6 16:48 tmp
```

{ #fig:010 width=70% }

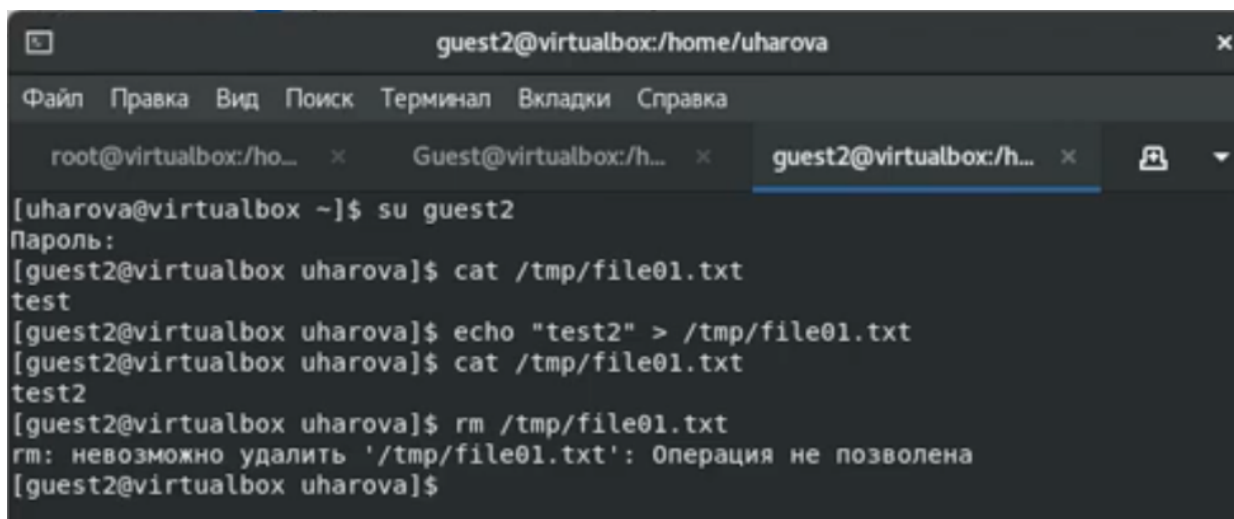
Создадим файл file1 и и разрешим чтение и запись для категории пользователей "все остальные"

```
[uharova@virtualbox ~]$ su Guest
Пароль:
[Guest@virtualbox uharova]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[Guest@virtualbox uharova]$ echo "test" > /tmp/file01.txt
[Guest@virtualbox uharova]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 Guest Guest 5 фев  6 16:52 /tmp/file01.txt
[Guest@virtualbox uharova]$
```

{ #fig:011 width=70% }

Пробуем записать информацию в file1

Удалить файл не получилось.



```
guest2@virtualbox:/home/uharova
Файл  Правка  Вид  Поиск  Терминал  Вкладки  Справка
root@virtualbox:/ho...  Guest@virtualbox:/h...  guest2@virtualbox:/h...
[uharova@virtualbox ~]$ su guest2
Пароль:
[guest2@virtualbox uharova]$ cat /tmp/file01.txt
test
[guest2@virtualbox uharova]$ echo "test2" > /tmp/file01.txt
[guest2@virtualbox uharova]$ cat /tmp/file01.txt
test2
[guest2@virtualbox uharova]$ rm /tmp/file01.txt
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена
[guest2@virtualbox uharova]$
```

{ #fig:012 width=70% }

Снимим атрибут -t

```
[guest2@virtualbox uharova]$ sudo su -  
  
Мы полагаем, что ваш системный администратор изложил вам основы  
безопасности. Как правило, всё сводится к трём следующим правилам:  
  
№1) Уважайте частную жизнь других.  
№2) Думайте, прежде что-то вводить.  
№3) С большой властью приходит большая ответственность.  
  
[sudo] пароль для guest2:  
[root@virtualbox ~]# chmod -t /tpm  
chmod: невозможно получить доступ к '/tpm': Нет такого файла или каталога  
[root@virtualbox ~]# chmod -t /tmp  
[root@virtualbox ~]# exit  
выход  
[guest2@virtualbox uharova]$
```

{ #fig:013 width=70% }

Повторим наши действия с file1

Повторили все действия и, в отличие от предыдущего раза, теперь уже нам удалось удалить файл:

```
[guest2@virtualbox uharova]$ rm /tmp/file01.txt  
[guest2@virtualbox uharova]$
```

{ #fig:014 width=70% }

Обратно вернем атрибут -t

```
[guest2@virtualbox uharova]$ sudo su -  
[root@virtualbox ~]# chmod +t /tmp  
[root@virtualbox ~]#
```

{ #fig:015 width=70% }

Вывод

В результате проделанной работы мы изучили механизмы изменения идентификаторов, применения SetUID- Sticky-битов, а так же получили практические навыки работы в консоли с дополнительными атрибутами.

