

Отчёт по лабораторной работе №5

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

Федорина Эрнест Василевич

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выводы	13
	Список литературы	14

Список иллюстраций

3.1	проверка наличия gss в системе	7
3.2	работа с simpleid.c	8
3.3	код в simpleid2.c	8
3.4	запуск simpleid2	8
3.5	работа с атрибутами файла simpleid	9
3.6	повторение действий с помощью SetGID	9
3.7	код readfile.c	10
3.8	работа с доступами и владельцами readfile.c	11
3.9	работа со Sticky битом	12

Список таблиц

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 Теоретическое введение

`chmod` (от англ. *change mode*) — команда для изменения прав доступа к файлам и каталогам, используемая в Unix-подобных операционных системах. Входит в стандарт POSIX, в Coreutils.[1].

3 Выполнение лабораторной работы

Для начала мы проверим наличие gcc и сделаем так, чтобы система защиты SELinux не мешала выполнению работы (рис. [3.1])

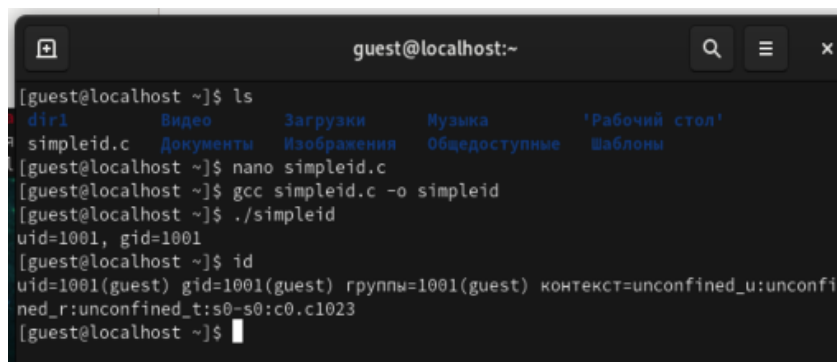
```
[root@localhost guest]# yum install gcc
Обновление репозитория службы управления подписками.
Невозможно прочитать идентификатор клиента

This system is not registered with an entitlement server. You can use "rhc" or "
subscription-manager" to register.

Последняя проверка окончания срока действия метаданных: 0:06:58 назад, Сб 28 сен
 2024 22:10:19.
Пакет gcc-11.5.0-2.el9.aarch64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[root@localhost guest]# setenforce 0
[root@localhost guest]# getenforce
Permissive
```

Рис. 3.1: проверка наличия gcc в системе

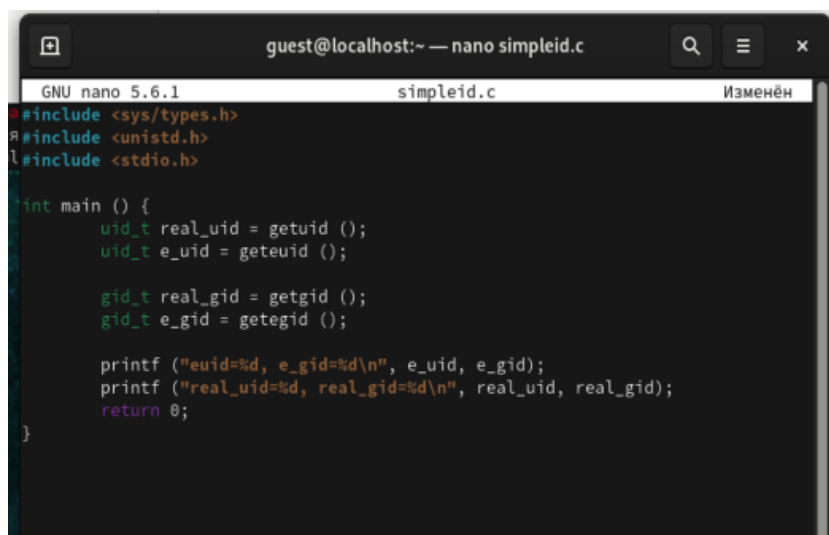
Создадим файл simpleid.c, напишем код в нём, а далее скомпилируем и запустим, сравним с выполнением команды id. Наш файл работает идентично и выдаёт ID пользователя (рис. [3.2])



```
guest@localhost:~  
[guest@localhost ~]$ ls  
dir1 Видео Загрузки Музыка 'Рабочий стол'  
simpleid.c Документы Изображения Общедоступные Шаблоны  
[guest@localhost ~]$ nano simpleid.c  
[guest@localhost ~]$ gcc simpleid.c -o simpleid  
[guest@localhost ~]$ ./simpleid  
uid=1001, gid=1001  
[guest@localhost ~]$ id  
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi  
ned_r:unconfined_t:s0-s0:c0.c1023  
[guest@localhost ~]$
```

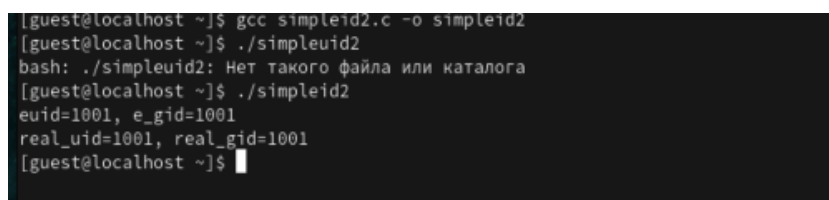
Рис. 3.2: работа с simpleid.c

Изменим код, добавив вывод дополнительных идентификаторов и запустим программу (рис. [3.3], [3.4])



```
GNU nano 5.6.1 simpleid.c Изменён  
#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 ("euid=%d, e_gid=%d\n", e_uid, e_gid);  
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);  
    return 0;  
}
```

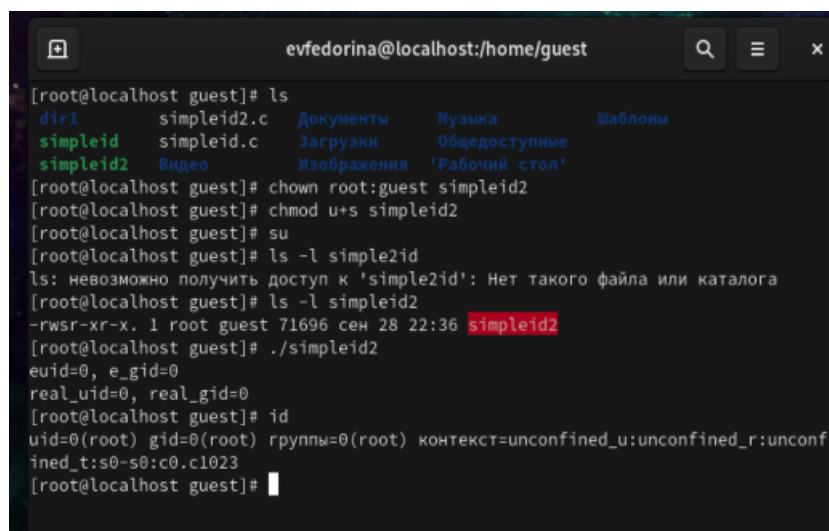
Рис. 3.3: код в simpleid2.c



```
[guest@localhost ~]$ gcc simpleid2.c -o simpleid2  
[guest@localhost ~]$ ./simpleid2  
bash: ./simpleid2: Нет такого файла или каталога  
[guest@localhost ~]$ ./simpleid2  
euid=1001, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@localhost ~]$
```

Рис. 3.4: запуск simpleid2

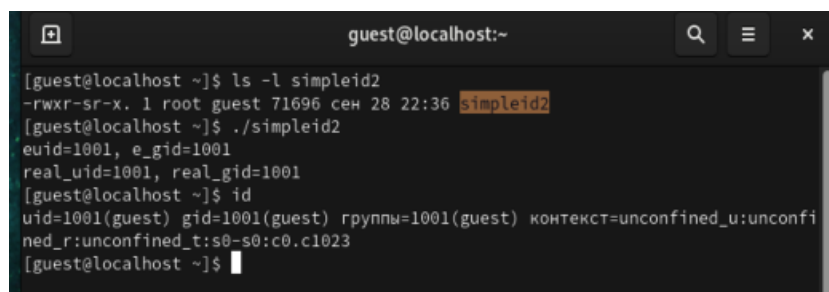
Выполним несколько команд от имени суперпользователя, проверим правильность установки новых атрибутов и смены владельца файла, а также ещё раз запустим файл и сравним его работу с командой `id`. Выдают идентичные результаты (рис. [3.5])



```
evfedorina@localhost:/home/guest
[root@localhost guest]# ls
dir1      simpleid2.c  Документы  Музыка      Шаблоны
simpleid   simpleid.c   Загрузки  Общедоступные
simpleid2  Видео       Изображения 'Рабочий стол'
[root@localhost guest]# chown root:guest simpleid2
[root@localhost guest]# chmod u+s simpleid2
[root@localhost guest]# su
[root@localhost guest]# ls -l simpleid2
ls: невозможно получить доступ к 'simpleid2': Нет такого файла или каталога
[root@localhost guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 71696 сен 28 22:36 simpleid2
[root@localhost guest]# ./simpleid2
euid=0, e_gid=0
real_uid=0, real_gid=0
[root@localhost guest]# id
uid=0(root) gid=0(root) rгруппы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost guest]#
```

Рис. 3.5: работа с атрибутами файла `simpleid`

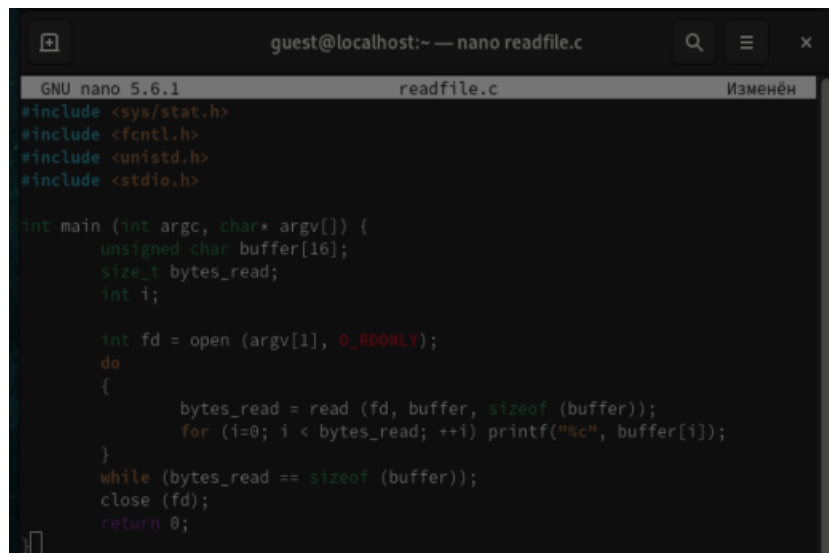
Прделаем то же самое с помощью SetGID-бита (рис. [3.6])



```
guest@localhost:~
[guest@localhost ~]$ ls -l simpleid2
-rwxr-sr-x. 1 root guest 71696 сен 28 22:36 simpleid2
[guest@localhost ~]$ ./simpleid2
euid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) rгруппы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost ~]$
```

Рис. 3.6: повторение действий с помощью SetGID

Напишем код для файла `readfile.c` (рис. [3.7])

A screenshot of a terminal window with a dark background. The title bar at the top shows 'guest@localhost:~ — nano readfile.c'. The editor header indicates 'GNU nano 5.6.1' and 'readfile.c' with a status 'Изменён'. The code is written in C and includes headers for `<sys/stat.h>`, `<fcntl.h>`, `<unistd.h>`, and `<stdio.h>`. The `main` function takes `argc` and `argv` as arguments. It declares a `buffer` of size 16, a `bytes_read` variable, and an index `i`. It opens a file at `argv[1]` in read-only mode (`O_RDONLY`). A `do-while` loop reads data from the file into the buffer and prints each character. The loop continues until `bytes_read` is less than the `sizeof` of the buffer. Finally, it closes the file and returns 0.

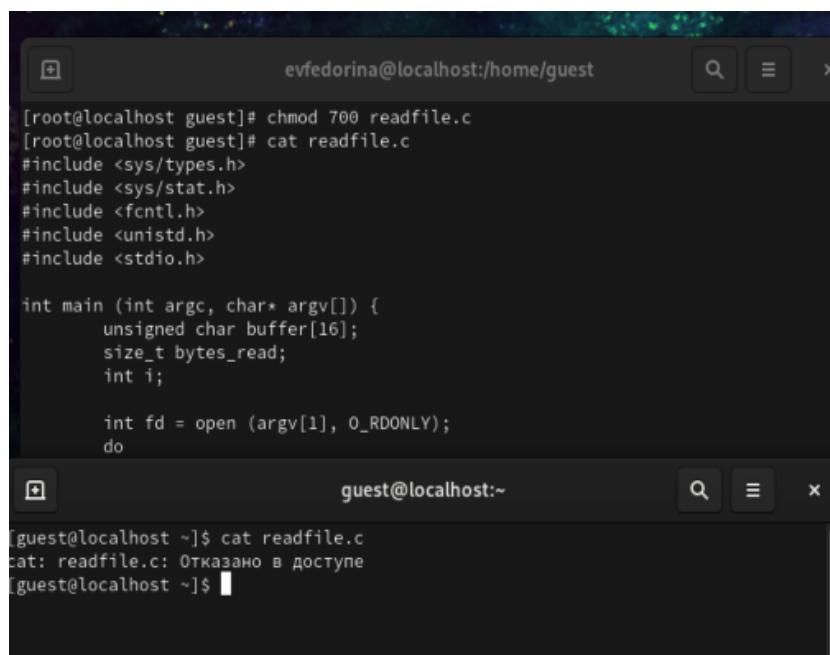
```
guest@localhost:~ — nano readfile.c
GNU nano 5.6.1 readfile.c Изменён
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

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. 3.7: код readfile.c

Сменим владельца, чтобы прочитать файл мог только суперпользователь. Попробуем прочитать файл от имени guest. Отказывает в доступе. Далее сменим владельца и установим SetUID-бит, проверим чтение файла, получилось (рис. [3.8])



The image shows two terminal windows. The top window is titled 'evfedorina@localhost:/home/guest' and shows a root user performing operations on a file named 'readfile.c'. The bottom window is titled 'guest@localhost:~' and shows a guest user attempting to view the same file, which results in a permission error.

```
[root@localhost guest]# chmod 700 readfile.c
[root@localhost guest]# cat readfile.c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.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
```

```
[guest@localhost ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@localhost ~]$
```

Рис. 3.8: работа с доступами и владельцами readfile.c

Исследуем Sticky-бит (рис. [3.9])

```
evfedorina@localhost [root@localhost guest]# cd tmp
[root@localhost tmp]# cd ..
[root@localhost guest]# chmod -t /tmp
[root@localhost guest]# chmod +t /tmp
[root@localhost guest]#

[guest2@localhost tmp]$ echo "test3" > file01.txt
bash: file01.txt: Отказано в доступе
[guest2@localhost tmp]$ echo "test2" >> file01.txt
bash: file01.txt: Отказано в доступе
[guest2@localhost tmp]$ echo "test3" > file01.txt
bash: file01.txt: Отказано в доступе
[guest2@localhost tmp]$ rm file01.txt
rm: удалить защищённый от записи пустой обычный файл 'file01.txt'? y
rm: невозможно удалить 'file01.txt': Отказано в доступе
[guest2@localhost tmp]$ su -
Пароль:
Последний вход в систему: С6 сен 28 22:39:56 MSK 2024 на pts/0
[root@localhost ~]# cd guest
-bash: cd: guest: Нет такого файла или каталога
[root@localhost ~]# cd tmp
-bash: cd: tmp: Нет такого файла или каталога
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# cd ~
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# cd
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# cd home
-bash: cd: home: Нет такого файла или каталога
[root@localhost ~]# cd guest
-bash: cd: guest: Нет такого файла или каталога
[root@localhost ~]# su guest
[guest@localhost root]$ cd ..
[guest@localhost /]$ cd guest
bash: cd: guest: Нет такого файла или каталога
[guest@localhost /]$ ls
bin boot etc lib media opt root sbin sys usr
[guest@localhost /]$ cd ..
[guest@localhost /]$ cd home
[guest@localhost home]$ ls
evfedorina guest guest2
[guest@localhost home]$ cd guest
[guest@localhost ~]$ cd tmp
[guest@localhost tmp]$ su guest2
Пароль:
[guest2@localhost tmp]$ ls -l / | grep tmp
drwxrwxrwx. 15 root root 4096 сен 28 23:08 tmp
[guest2@localhost tmp]$ cat file01.txt
[guest2@localhost tmp]$ echo "test2" >> file01.txt
bash: file01.txt: Отказано в доступе
[guest2@localhost tmp]$ echo "test3" > file01.txt
bash: file01.txt: Отказано в доступе
[guest2@localhost tmp]$
```

Рис. 3.9: работа со Sticky битом

4 Выводы

Изучил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получил практические навыки работы в консоли с дополнительными атрибутами. Рассмотрел работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы

1. chmod [Электронный ресурс]. Wikimedia Foundation, Inc., 2024. URL: <https://en.wikipedia.org/wiki/Chmod>.