Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Маслова Анастасия Сергеевна

Содержание

[1 Цель работы 1](#_Toc179041187)

[2 Выполнение лабораторной работы 1](#_Toc179041188)

[2.1 Создание программы 1](#_Toc179041189)

[2.2 Исследование Sticky-бита 6](#_Toc179041190)

[3 Выводы 7](#_Toc179041191)

[Список литературы 7](#_Toc179041192)

# 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.

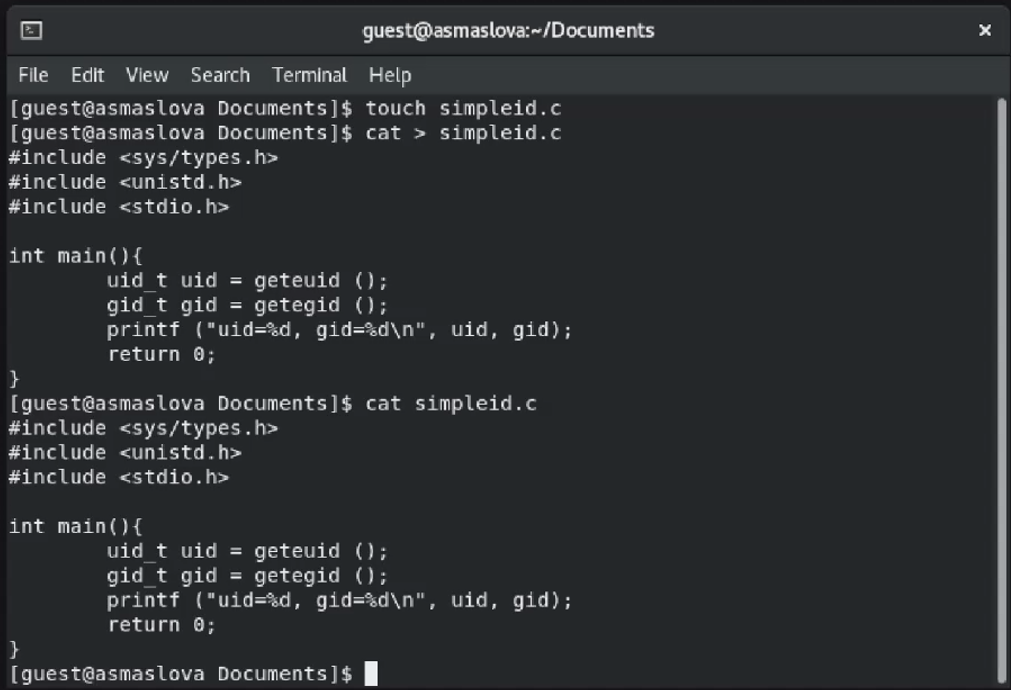
# 2 Выполнение лабораторной работы

## 2.1 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;  
}

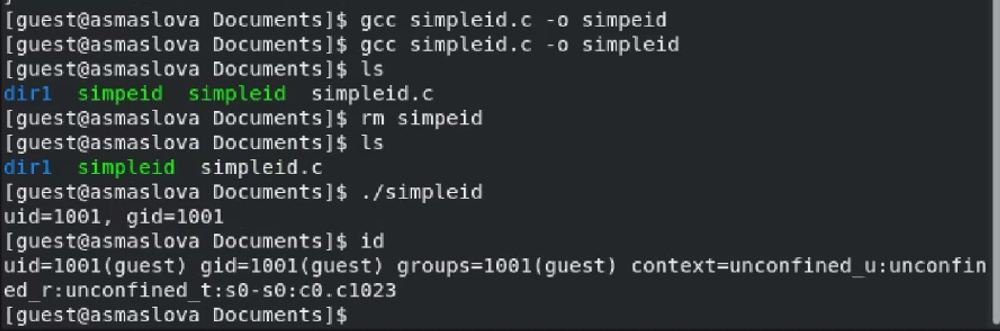
Выполнение этих двух пунктов вы можете видеть ниже (рис. [??]).



Создание программы simpleid.c

1. Скомплилируйте программу и убедитесь, что файл программы создан: gcc simpleid.c -o simpleid
2. Выполните программу simpleid: ./simpleid
3. Выполните системную программу id: id и сравните полученный вами результат с данными предыдущего пункта задания.

Выполнение этих трех пунктов вы можете видеть ниже (рис. [??]).



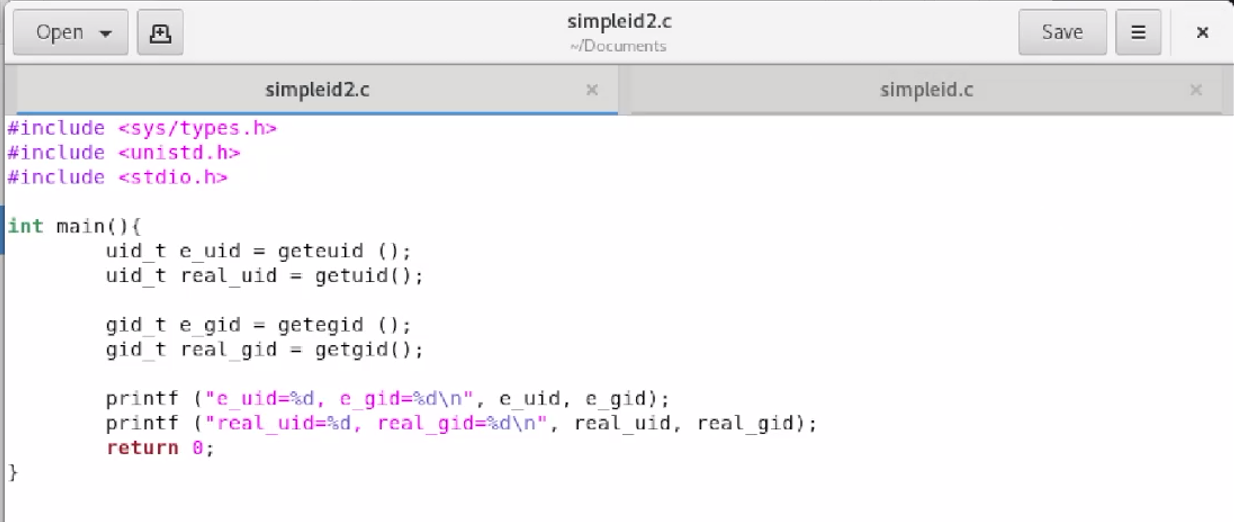
Компиляция и выполнение программы simpleid.c

1. 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.

Я создала файл simpleid2.c и записала туда этот код (рис. [??]).

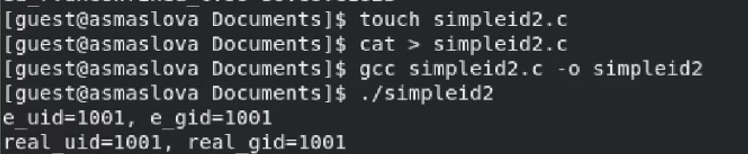


Создание программы simpleid2.c

1. Скомпилируйте и запустите simpleid2.c:

gcc simpleid2.c -o simpleid2  
./simpleid2

Я скомпилировала и запустила названный файл (рис. [??]).



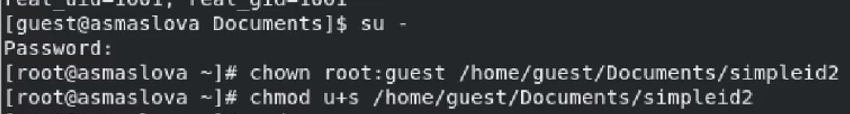
Компиляция и выполнение программы simpleid2.c

1. От имени суперпользователя выполните команды:

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

1. Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды.

Повысив свои права с помощью команды su -, я от имени суперпользователя выполнила указанные команды для изменения прав на файл simpleid2 (рис. [??]).

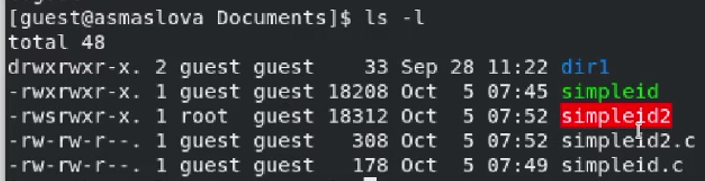


Изменение прав на файл simpleid2 от имени суперпользователя

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

ls -l simpleid2

Выполнение этого пункта вы можете видеть ниже (рис. [??]).



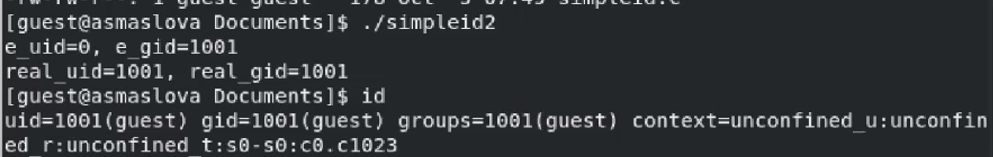
Проверка правильности установки новых атрибутов

1. Запустите simpleid2 и id:

./simpleid2  
id

Сравните результаты.

Выполнение этого пункта вы можете видеть ниже (рис. [??]).



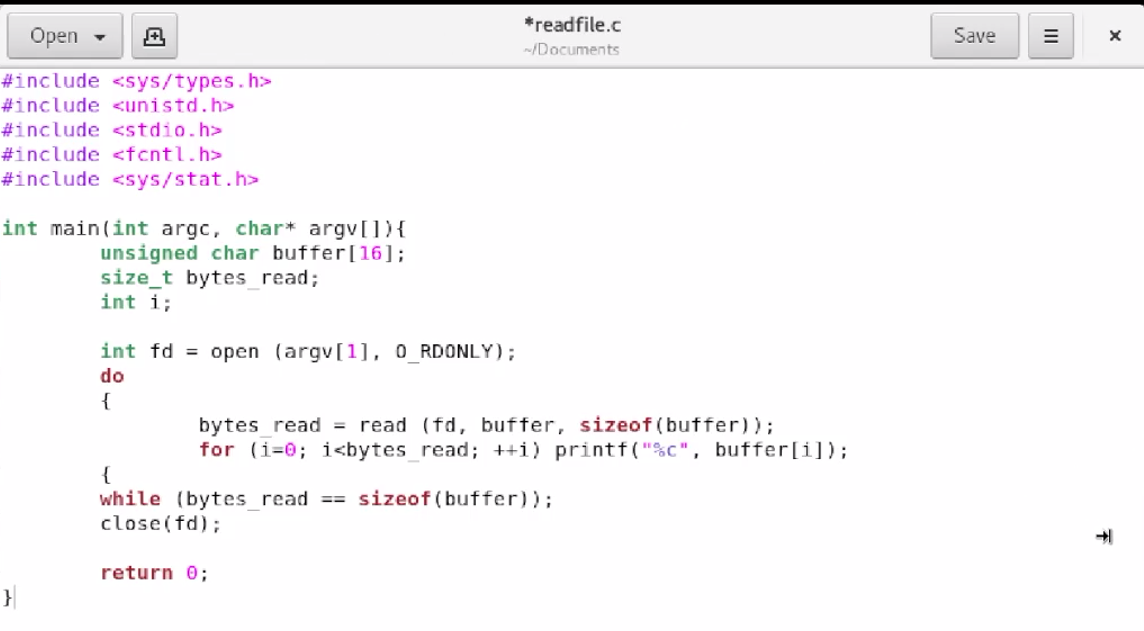
Компиляция и выполнение программы simpleid2.c

По сравнению с работой программы simpleid, в выводе simpleid2 присутствовали отличающиеся данные, а конкретно - e\_uid=0.

1. Проделайте тоже самое относительно SetGID-бита.
2. 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;  
}

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

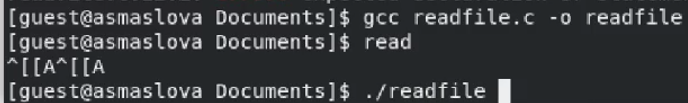


Создание программы readfile.c

1. Откомпилируйте её.

gcc readfile.c -o readfile

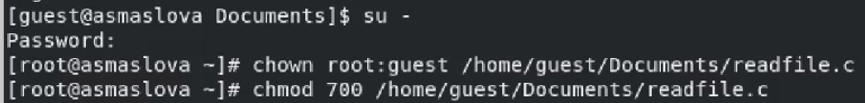
Выполнение этого пункта вы можете видеть ниже (рис. [??]).



Компиляция и выполнение программы readfile.c

1. Смените владельца у файла readfile.c (или любого другого текстового файла в системе) и измените права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог.

Выполнение этого пункта вы можете видеть ниже (рис. [??]).



Смена владельца файла readfile.c

1. Проверьте, что пользователь guest не может прочитать файл readfile.c.

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

Невозможность прочитать файл readfile.c

Невозможность прочитать файл readfile.c

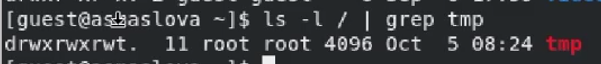
1. Смените у программы readfile владельца и установите SetU’D-бит.
2. Проверьте, может ли программа readfile прочитать файл readfile.c?
3. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отразите полученный результат и ваши объяснения в отчёте.

## 2.2 Исследование Sticky-бита

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

ls -l / | grep tmp

Выполнение этого пункта вы можете видеть ниже (рис. [??]). Как можно заметить, атрибут установлен, потому что в конце стоит буква t.



Проверка наличия атрибута Sticky на директории /tmp

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

echo "test" > /tmp/file01.txt

1. 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

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

|  |
| --- |
| Просмотр атрибутов |

Просмотр атрибутов

1. От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt: cat /tmp/file01.txt
2. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой echo "test2" > /tmp/file01.txt. Удалось ли вам выполнить операцию?

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

|  |
| --- |
| Попытка дозаписать в файл |

Попытка дозаписать в файл

1. Проверьте содержимое файла командой cat /tmp/file01.txt

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

|  |
| --- |
| Проверка содержимого файла |

Проверка содержимого файла

1. 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?
2. Проверьте содержимое файла командой cat /tmp/file01.txt
3. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой rm /tmp/fileOl.txt. Удалось ли вам удалить файл?
4. Повысьте свои права до суперпользователя следующей командой su - и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp
5. Покиньте режим суперпользователя командой exit
6. От пользователя guest2 проверьте, что атрибута t у директории /tmp нет: ls -l / | grep tmp

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

|  |
| --- |
| Проверка отсутствия атрибута Sticky на директории /tmp |

Проверка отсутствия атрибута Sticky на директории /tmp

1. Повторите предыдущие шаги. Какие наблюдаются изменения?

Единственное изменение, которое я заметила - это тот факт, что с этим атрибутом я могу удалить файл readfile.c.

1. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Ваши наблюдения занесите в отчёт.
2. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp:

su -  
chmod +t /tmp  
exit

# 3 Выводы

В ходе лабораторной работы я изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов, получила практические навыки работы в консоли с дополнительными атрибутами, рассмотрела работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# Список литературы