Лабораторная работа №5

Дисциплина: Информационная безопасность

Губина Ольга Вячеслвовна

Содержание

# 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 Задание

* Изучить на практике применение SetUID- и Sticky-битов.

# 3 Теоретическое введение

**Setuid** – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволяет нам поднять привилегии пользователя в случае, если это необходимо. Классический пример использования этого бита в операционной системе это команда sudo.

Там, где обычно установлен классический бит **x** (на исполнение), выставляется специальный бит **s**. Это позволяет обычному пользователю системы выполнять команды с повышенными привилегиями без необходимости входа в систему как root, разумеется зная пароль пользователя root. Принцип работы **Setgid** очень похож на setuid с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

Биты **setuid** и **setgid** выставляются с помощью команд chmod u+s и chmod g+s соответственно.[1]

**Sticky-bit** — дополнительный атрибут файлов или каталогов в операционных системах семейства UNIX. В настоящее время sticky bit используется в основном для каталогов, чтобы защитить в них файлы. Из такого каталога пользователь может удалить только те файлы, владельцем которых он является. Примером может служить каталог /tmp, в который запись открыта для всех пользователей, но нежелательно удаление чужих файлов. Установка атрибута производится утилитой chmod.

В операционной системе Solaris для файлов, не являющихся программами, имеет строго противоположное действие — запрещает сохранение данных этого файла в системном кэше.[2]

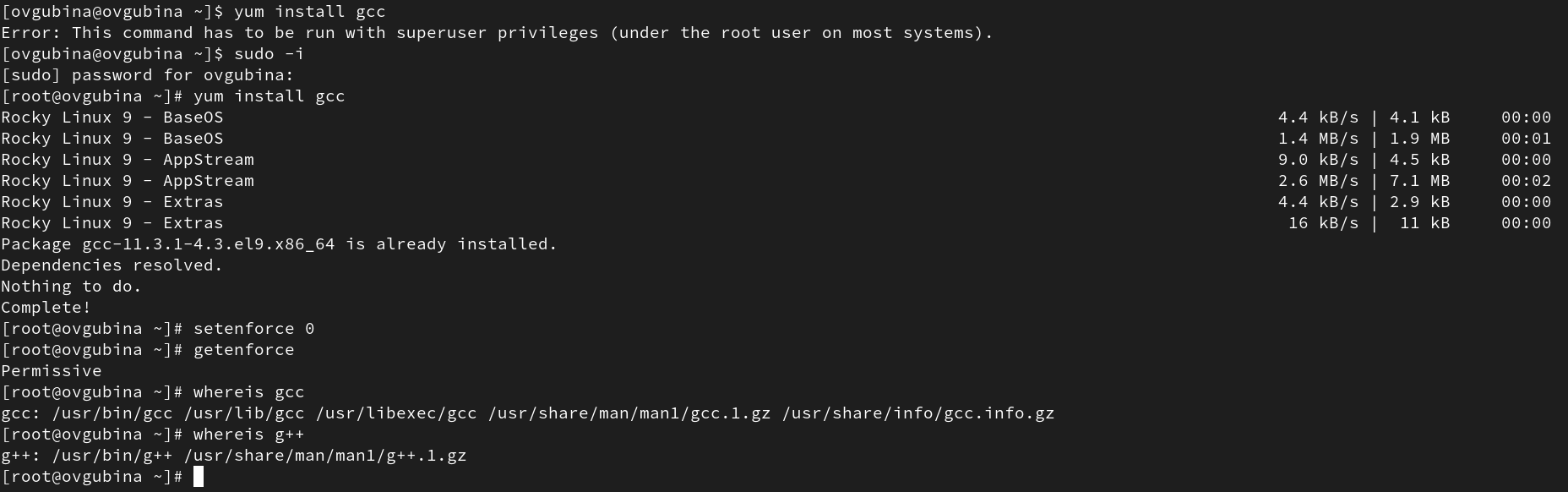
Это разрешение полезно для защиты файлов от случайного удаления в среде, где несколько пользователей имеют права на запись в один и тот же каталог. Если применяется закрепленный sticky bit, пользователь может удалить файл, только если он является пользователем-владельцем файла или каталога, в котором содержится файл. По этой причине он применяется в качестве разрешения по умолчанию для каталога /tmp и может быть полезен также для каталогов общих групп.

Когда вы применяете sticky bit, пользователь может удалять файлы, только если выполняется одно из следующих условий:

* Пользователь является владельцем файла;
* Пользователь является владельцем каталога, в котором находится файл.[3]

# 4 Выполнение лабораторной работы

Прежде чем начать выполнять работу, проверим, имеются ли на устройстве все необходимые ресурсы для осуществления компиляции (рис. ??).

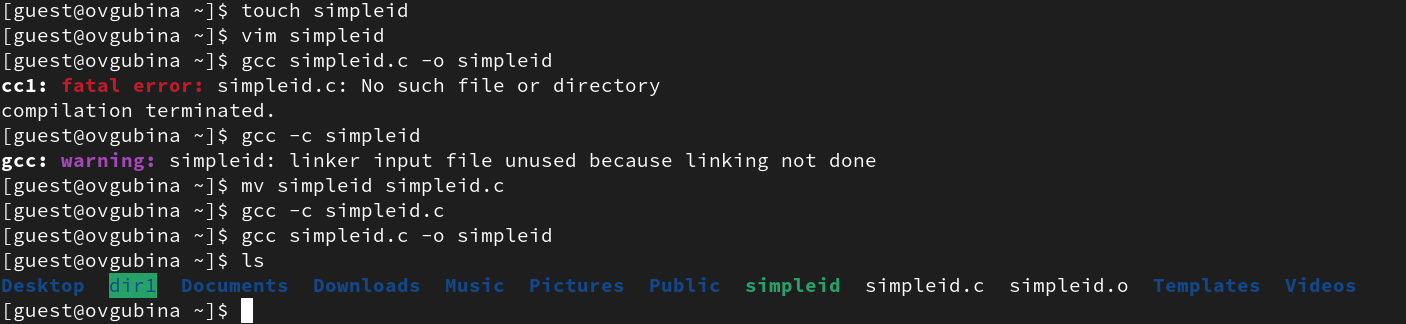


Проверка необходимых ресурсов

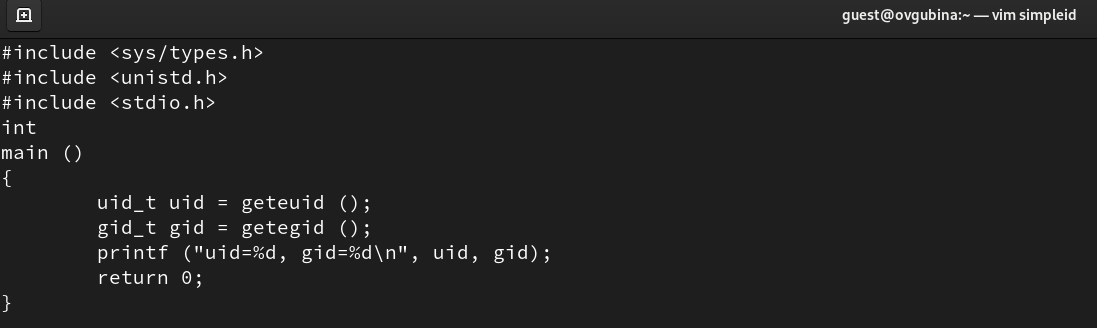
## 4.1 Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте программу simpleid.c (рис. ??-??):

#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;  
}



Создание и компиляция файла



Программа simpleid.c

1. Скомплилируйте программу и убедитесь, что файл программы создан: gcc simpleid.c -o simpleid (рис. ??).

Видим, что никаких ошибок при компиляции не возникает, файлы успешно созданы (проверка через команду ls).

1. Выполните программу simpleid (рис. ??): ./simpleid.



Вызов программы simpleid.c

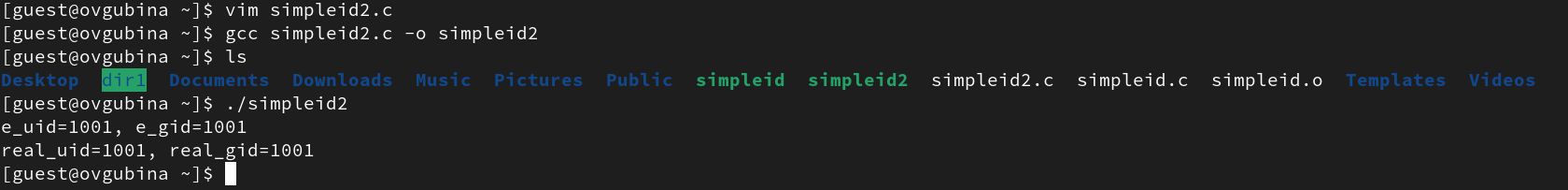
1. Выполните системную программу id (рис. ??): id. и сравните полученный вами результат с данными предыдущего пункта задания.

Видим, что выводы программы, написанной ранее, и команды id совпадают - в каждой из них предоставляется информация об id владельца и группы, ими является guest с uid = 1001, gid = 1001.

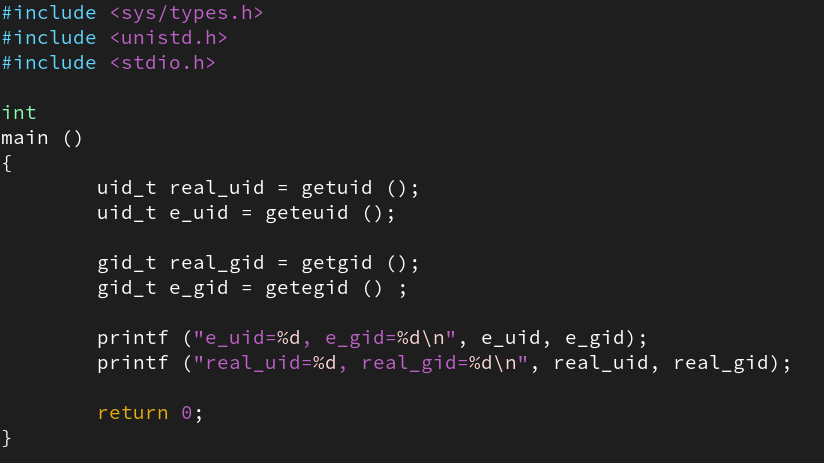
1. Усложните программу, добавив вывод действительных идентификаторов (рис. ??):

#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 (рис. ??).



Создание и компиляция simpleid2.c



Программа simpleid2.c

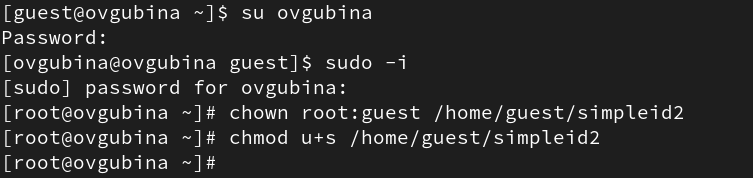
1. Скомпилируйте и запустите simpleid2.c (рис. ??):

gcc simpleid2.c -o simpleid2  
./simpleid2

Видим, что теперь выводятся айди владельца файла и айди текущего пользователя - сейчас это guest с uid = 1001, gid = 1001.

1. От имени суперпользователя выполните команды (рис. ??):

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



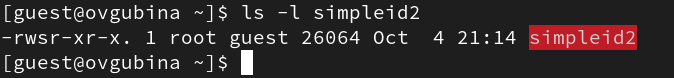
Добавление SETUID

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

В данном случае я использвала sudo в дополнительном терминале.

Команды отличаются следующим: *su* требует пароль целевой учетной записи (например, пользователя root) и переключает вас на нее, в то время как sudo требует пароль текущего пользователя и запускает от его имени только лишь одну (или несколько) команд, на выполнение которых требуются права суперпользователя.

1. Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2 (рис. ??): ls -l simpleid2

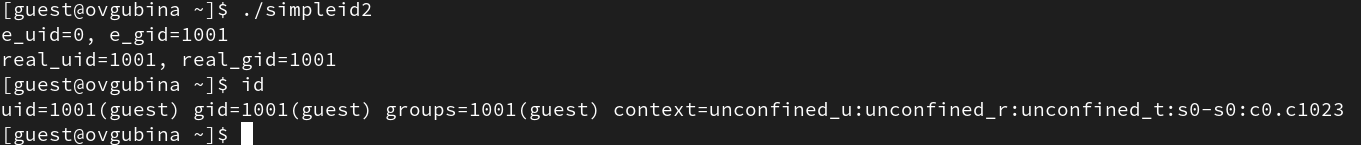


Проверка добавления SETUID-бита

Видим добавленный SetUID-бит с правах доступа пользователя-владельца - **s**.

1. Запустите simpleid2 и id:

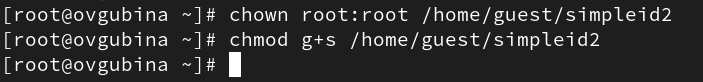
./simpleid2  
id



Выполнение программы после добавления SETUID-бита

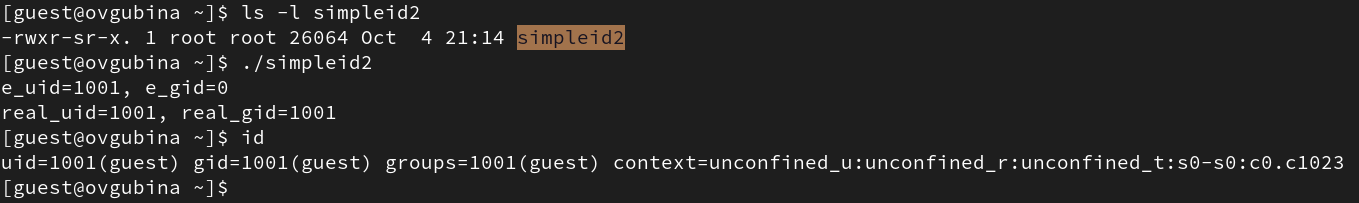
Видим, что теперь айди пользователя владельца - это айди пользователя root.

1. Проделайте тоже самое относительно SetGID-бита.



Добавление setGID-бита

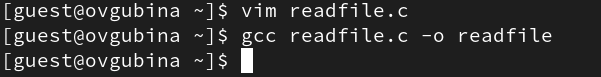
От пользвоателя guest смотрим назначение бита для группы - видим, что он добавился (рис. ??). Запускаем программу и видим теперь поменявшееся айди группы пользователей на айди root (рис. ??).



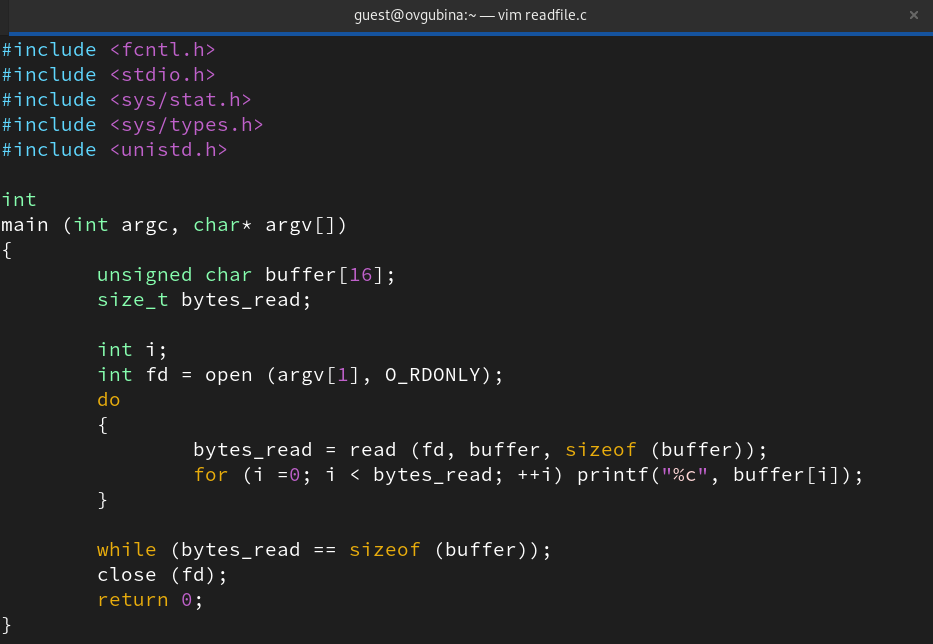
Выполнение программы после добавления SETUID-бита

1. Создайте программу readfile.c (рис. ??):

#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;  
}



Создание и компиляция программы readfile.c



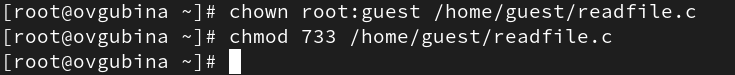
Программа readfile.c

1. Откомпилируйте её (рис. ??):

gcc readfile.c -o readfile

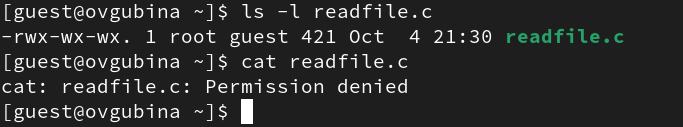
1. Смените владельца у файла readfile.c (или любого другого текстового файла в системе) и измените права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог.

Владельца файла меняем на root, назначаем права доступа 733 - дают полные права пользователю-владельцу и права на запись и выполнение остальным пользователям и группе (рис. ??).



Программа readfile.c

1. Проверьте, что пользователь guest не может прочитать файл readfile.c (рис. ??).

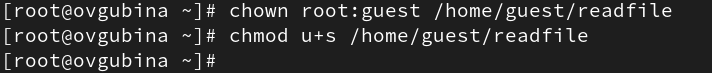


Проверка запрета на чтение файла

Пользователь действительно не может прочитать файл readfile.c - выводится сообщение **Permission denied**.

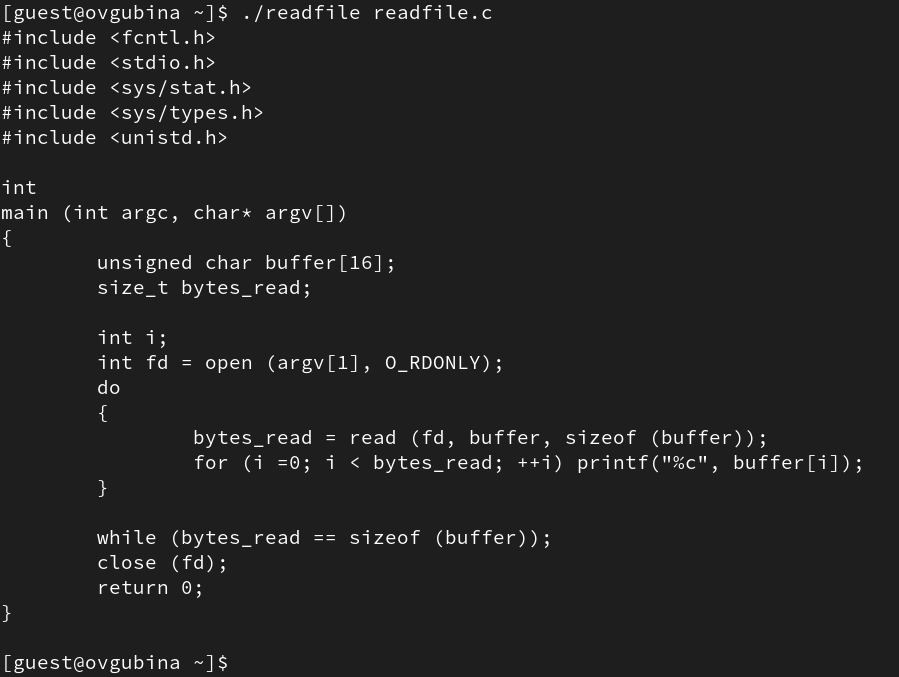
1. Смените у программы readfile владельца и установите SetUID-бит.

Меняем вадельца на root и назначаем SetUID-бит (рис. ??).



Назначение SetUID-бита

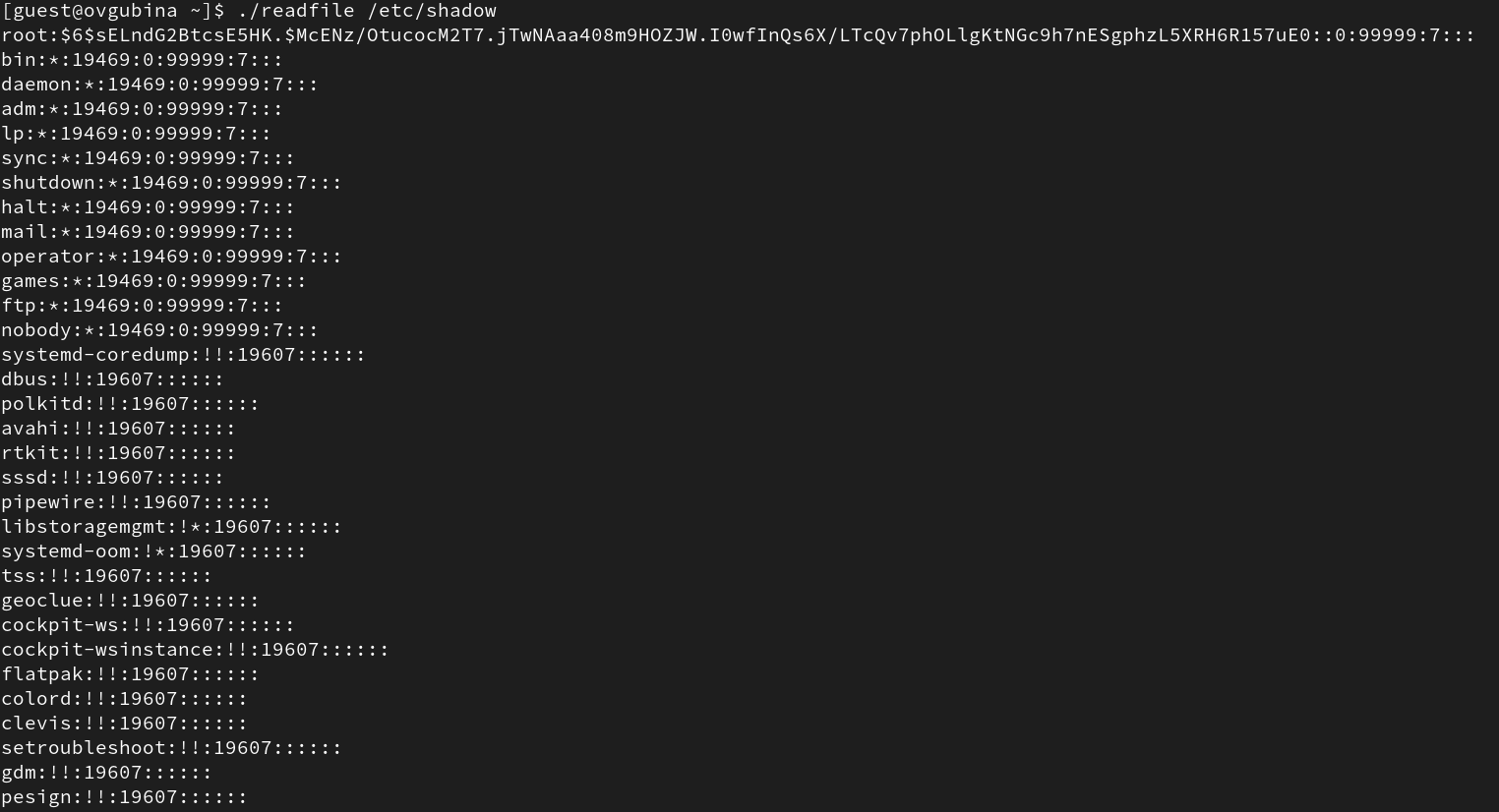
1. Проверьте, может ли программа readfile прочитать файл readfile.c (рис. ??)?



Попытка прочитать файл readfile.c через программу readfile

Нам удалось прочитать файл.

1. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отразите полученный результат и ваши объяснения в отчёте.

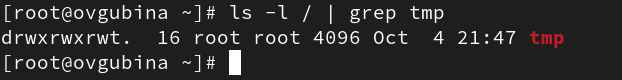


Попытка прочитать файл /etc/shadow через программу readfile

Нам удалось прочитать оба файла поскольку мы делали это с помощью программы readfile, которая принадлежит суперпользователю root.

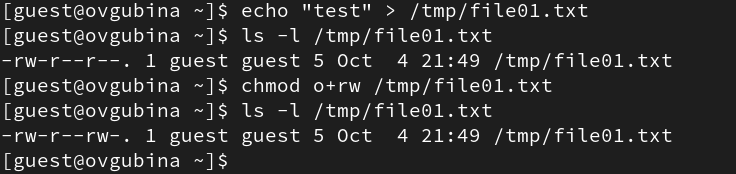
## 4.2 Исследование Sticky-бита

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду ls -l / | grep tmp (рис. ??).



Проверка наличия Sticky-бита

1. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: echo "test" > /tmp/file01.txt (рис. ??)

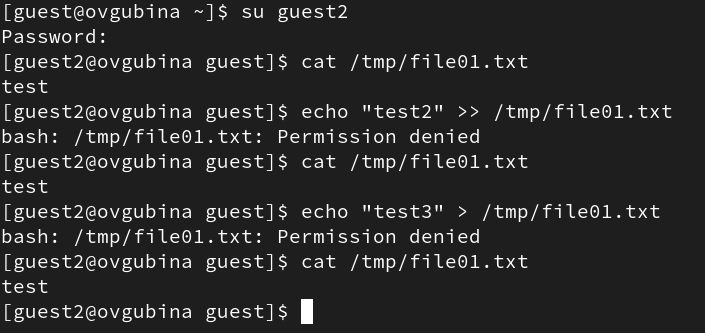


Создание и запись в файл file01.txt

1. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные» (рис. ??):

ls -l /tmp/file01.txt  
chmod o+rw /tmp/file01.txt  
ls -l /tmp/file01.txt

1. От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt: cat /tmp/file01.txt (рис. ??).



guest2 - чтение файла

Содержимое файла выводится - можем прочитать файл.

1. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой echo "test2" >> /tmp/file01.txt (рис. ??). Удалось ли вам выполнить операцию?

Выполнить операцию не удалось - Permission denied.

1. Проверьте содержимое файла командой cat /tmp/file01.txt (рис. ??).

Файл никак не изменился.

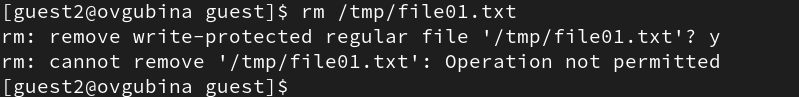
1. От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой echo "test3" > /tmp/file01.txt (рис. ??). Удалось ли вам выполнить операцию?

Выполнить операцию не удалось - Permission denied.

1. Проверьте содержимое файла командой cat /tmp/file01.txt (рис. ??).

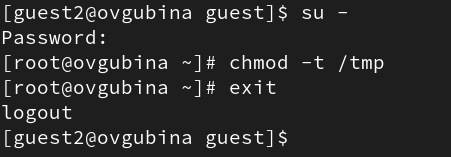
Файл никак не изменился.

1. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой rm /tmp/fileOl.txt (рис. ??). Удалось ли вам удалить файл?



Попытка удаления файла со Sticky-битом

1. Повысьте свои права до суперпользователя следующей командой su - и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp (рис. ??).



Снятие с директории Sticky-бита

1. Покиньте режим суперпользователя командой exit (рис. ??).
2. От пользователя guest2 проверьте, что атрибута **t** у директории /tmp нет: ls -l / | grep tmp (рис. ??)

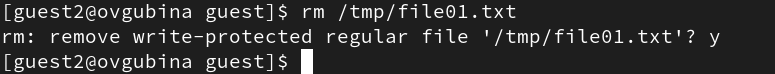


Проверка снятия с директории Sticky-бита

1. Повторите предыдущие шаги (рис. ??). Какие наблюдаются изменения?

Изменений не наблюдается, мы также можем читать файл, но не можем производить дозапись и запись в него.

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

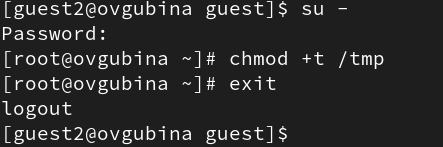


Попытка удаления файла без Sticky-бита

В жтот раз без Sticky-бита нам удалось успешно удалисть файл file01.txt.

1. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp (рис. ??):

su -  
chmod +t /tmp  
exit



Возвращение Sticky-бита

# 5 Выводы

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

# Список литературы

1. Использование SETUID, SETGID и Sticky bit для расширенной настройки прав доступа в операционных системах Linux [Электронный ресурс]. 2023. URL: <https://ruvds.com/ru/helpcenter/suid-sgid-sticky-bit-linux/>.

2. Sticky bit [Электронный ресурс]. 2023. URL: <https://ru.wikipedia.org/wiki/Sticky_bit>.

3. Права в Linux (chown, chmod, SUID, GUID, sticky bit, ACL, umask) [Электронный ресурс]. 2023. URL: <https://habr.com/ru/articles/469667/>.