Отчет по лаборатной работе №5

по предмету Информационная безопасность

Алхимова Дарья Сергеевна

Содержание

[Цель работы 1](#__RefHeading___Toc804_3682143584)

[Задание 1](#__RefHeading___Toc806_3682143584)

[Часть 1 1](#__RefHeading___Toc808_3682143584)

[Часть 2 3](#__RefHeading___Toc810_3682143584)

[Теоретическое введение 4](#__RefHeading___Toc812_3682143584)

[Выполнение лабораторной работы 5](#__RefHeading___Toc814_3682143584)

[Часть 1 5](#__RefHeading___Toc816_3682143584)

[Часть 2 11](#__RefHeading___Toc818_3682143584)

[Выводы 14](#__RefHeading___Toc820_3682143584)

[Список литературы 14](#__RefHeading___Toc822_3682143584)

# Цель работы

Целью данной работы является изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в кон- соли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# Задание

## Часть 1

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;  
  }

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.
4. Усложните программу, добавив вывод действительных идентификато- ров:

* #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.

1. Скомпилируйте и запустите simpleid2.c:  
   gcc simpleid2.c -o simpleid2  
   ./simpleid2
2. От имени суперпользователя выполните команды: chown root:guest /home/guest/simpleid2 chmod u+s /home/guest/simpleid2
3. Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды.
4. Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2: ls -l simpleid2
5. Запустите simpleid2 и id: ./simpleid2 id Сравните результаты.
6. Проделайте тоже самое относительно SetGID-бита.
7. 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;  
  }

1. Откомпилируйте её: gcc readfile.c -o readfile.
2. 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.
3. Проверьте, что пользователь guest не может прочитать файл readfile.c.
4. Смените у программы readfile владельца и установите SetU’D-бит.
5. Проверьте, может ли программа readfile прочитать файл readfile.c?
6. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отразите полученный результат и ваши объяснения в отчёте.

## Часть 2

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/fileOl.txt Удалось ли вам удалить файл?
10. Повысьте свои права до суперпользователя 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 ди- ректорию /tmp:

* su -  
   chmod +t /tmp  
   exit

# Теоретическое введение

* Компиляторы, доступные в Linux-системах, являются частью коллек- ции GNU-компиляторов, известной как GCC (GNU Compiller Collection, подробнее см. [http://gcc.gnu.org](http://gcc.gnu.org/)). В неё входят компиляторы языков С, С++, Java, Objective-C, Fortran и Chill. Будем использовать лишь первые два. Компилятор языка С называется gcc. Компилятор языка С++ называется g++ и запускается с параметрами почти так же, как gcc. Проверить это можно следующими командами:

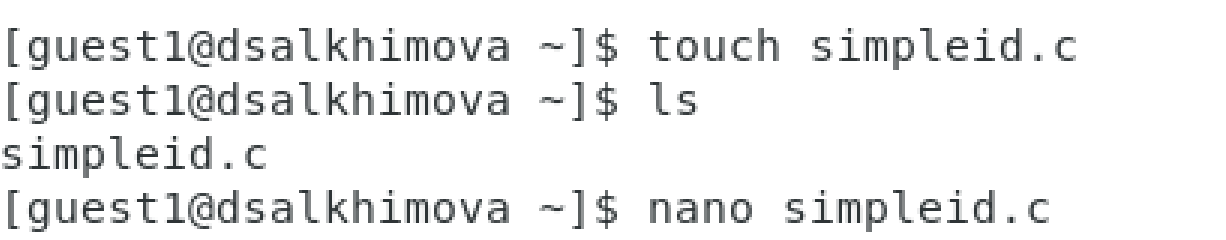
whereis gcc  
whereis g++

Первый шаг заключается в превращении исходных файлов в объектный код: gcc -c file.с В случае успешного выполнения команды (отсутствие ошибок в коде) полученный объектный файл будет называться file.о. Объектные файлы невозможно запускать и использовать, поэтому после компиляции для получения готовой программы объектные файлы необходимо скомпоновать. Компоновать можно один или несколько файлов. В случае использования хотя бы одного из файлов, написанных на С++, 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 С, 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 +х).

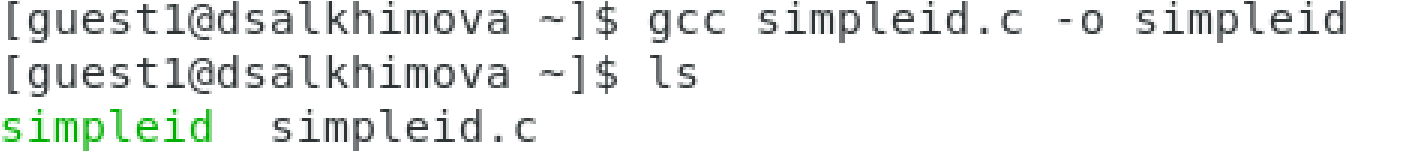
# Выполнение лабораторной работы

## Часть 1

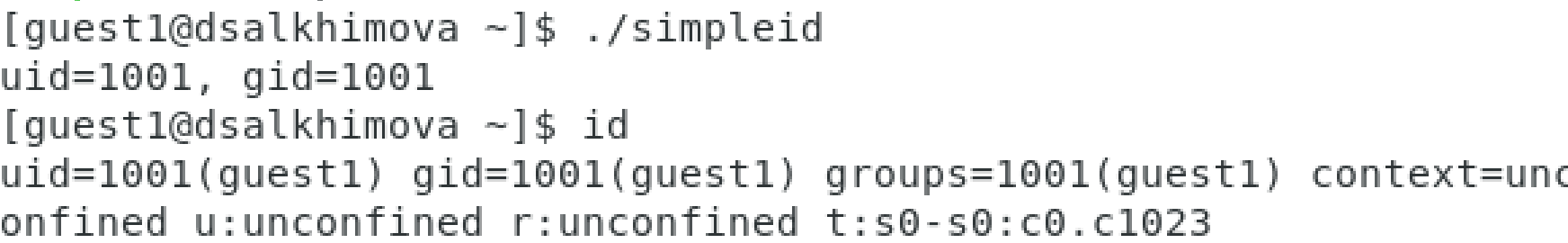
1. Вошла в систему от имени пользователя guest и создала программу simpleid.c. ([рис. 1](images5/1.png)). С помощью команды nano открыла файл для редактирования и внесла туда код из задания:

* #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 файла программы

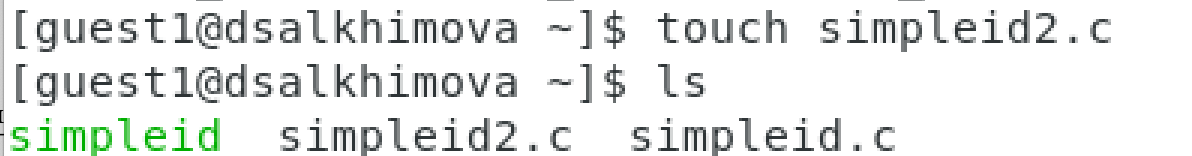
1. Скомплилировала программу и убедилась, что файл программы создан: gcc simpleid.c -o simpleid ([рис. 2](images5/2.png)).

* 
* Проверка наличия файла

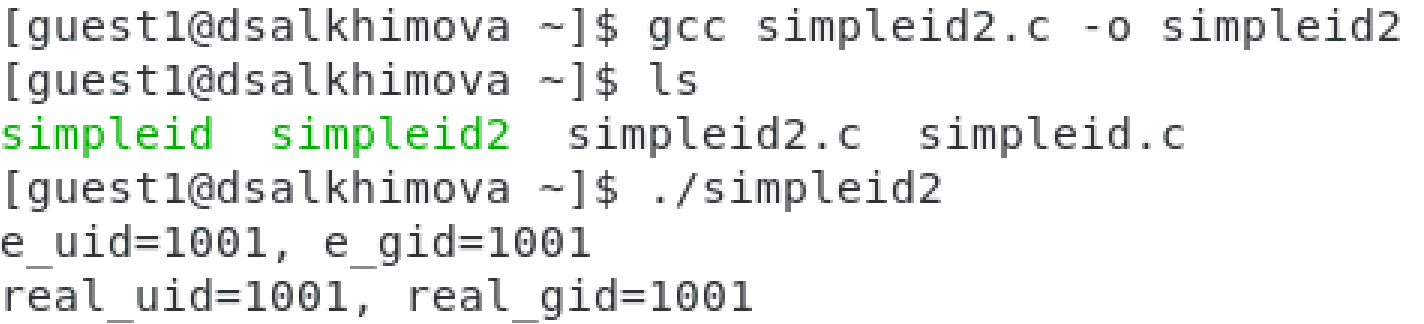
1. Выполнила программу simpleid (./simpleid) и системную программу id (id). Вывод программ сходится - они передают одинаковые параметры пользователя и его группы. ([рис. 3](images5/3.png)).

* 
* Сравнения вывода программ simpleid и id

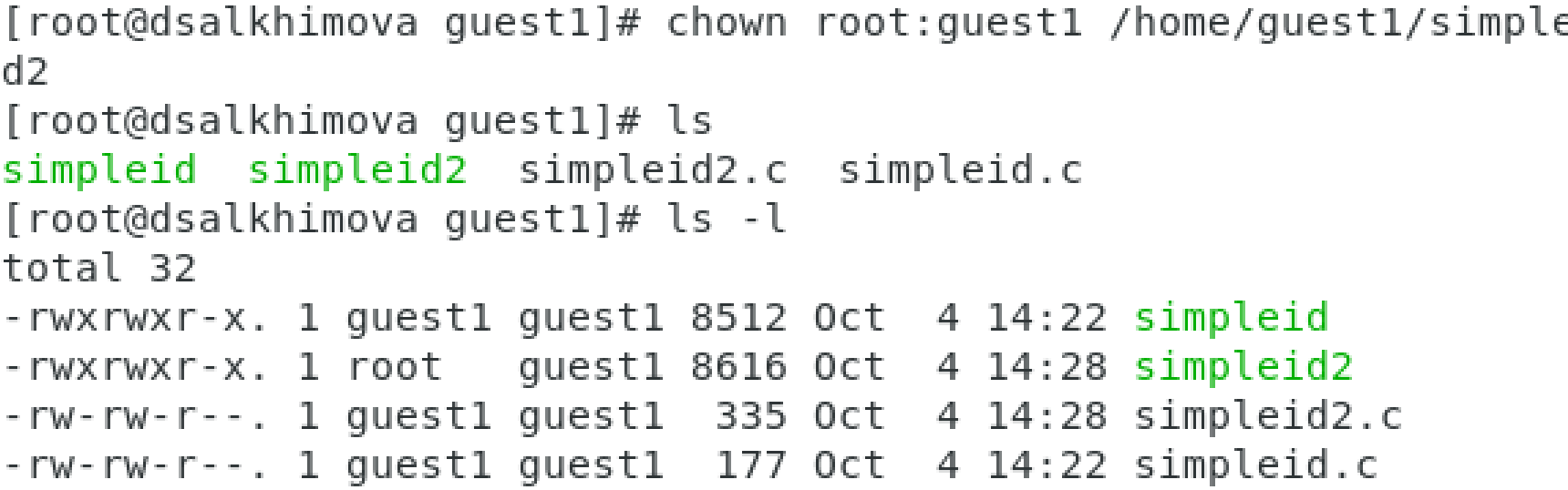
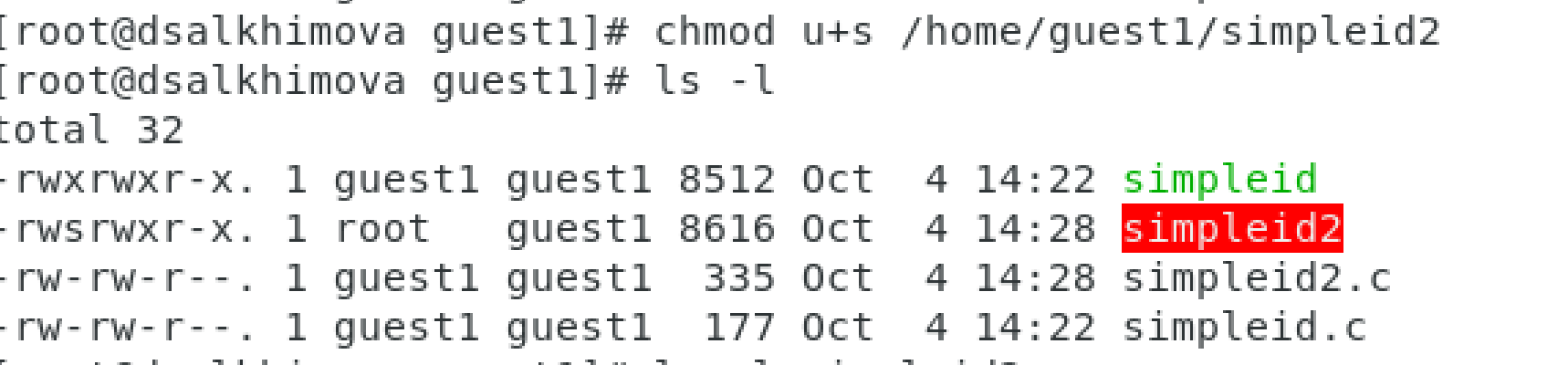
1. Усложнила программу, добавив вывод действительных идентификаторов: #include <sys/types.h>  
   #include <unistd.h>  
   #include <stdio.h>  
   int  
   main ()  
   {  
   uid\_t real\_uid = getuid (); uid\_t e\_uid = geteuid ();  
   gid\_t real\_gid = getgid (); gid\_t e\_gid = getegid ();  
   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 назвала simpleid2.c. ([рис. 4](images5/4.png)).

* 
* Добавление в программу действительных индентификаторов

1. Скомпилировала и запустила simpleid2.c ([рис. 5](images5/5.png)).: gcc simpleid2.c -o simpleid2 ./simpleid2

* 
* Компилирование и запуск программы simpleid2

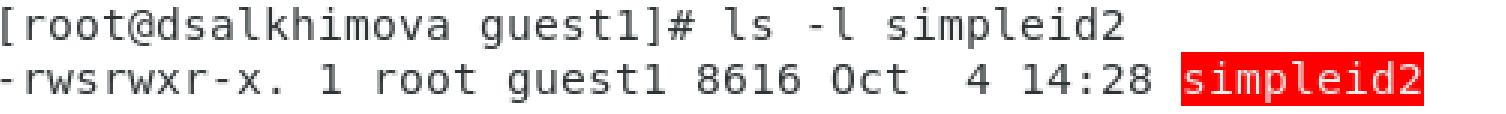
1. От имени суперпользователя выполнила команды:  
   chown root:guest /home/guest/simpleid2 ([рис. 6](images5/6.png)) и chmod u+s /home/guest/simpleid2 ([рис. 7](images5/7.png))

*   
  

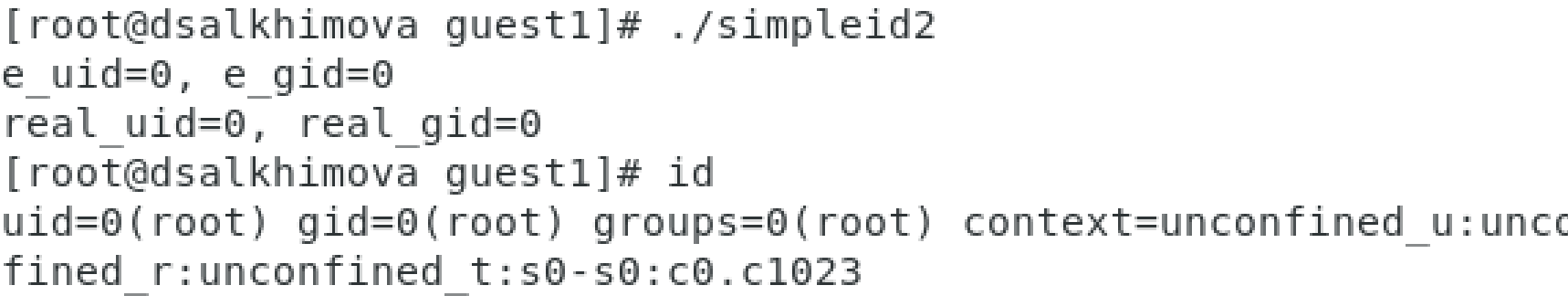
1. Повысила временно свои права с помощью su.

* Самый частый пример использования sudo - выполнение программы от имени суперпользователя. Для этого достаточно написать sudo перед именем программы.
* Если вызов команды su происходит без аргументов, то происходит смена пользователя оболочки shell на суперпользователя root. Программа выдаст приглашение ввода пароля, если пароль будет верным, то текущим пользователем станет root.

1. Выполнила проверку правильности установки новых атрибутов и смены владельца файла simpleid2: ls -l simpleid2 ([рис. 8](images5/8.png))

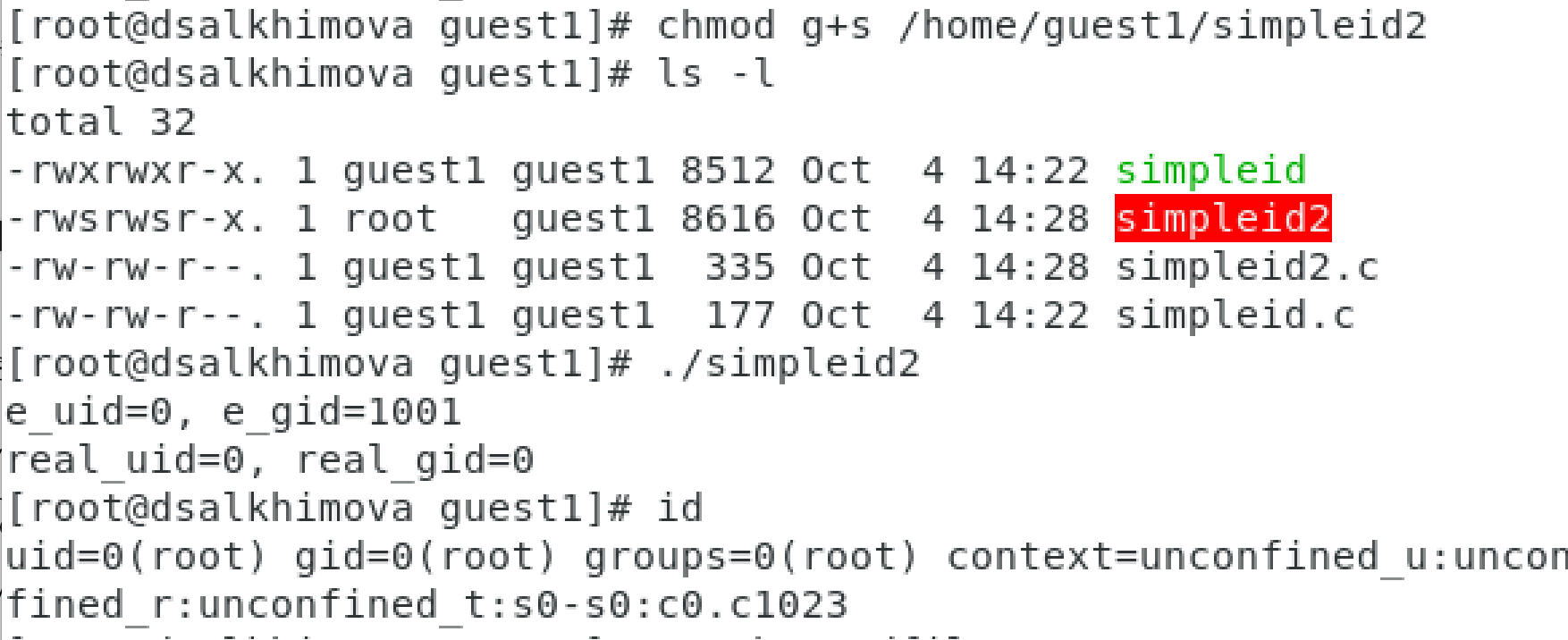
* 
* Проверка изменений файла

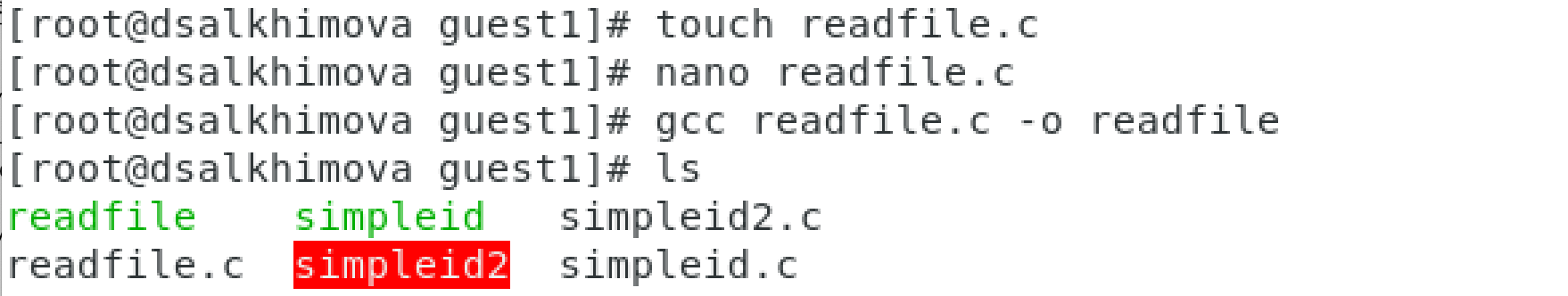
1. Запустила simpleid2 (./simpleid2) и id (id) ([рис. 9](images5/9.png)).

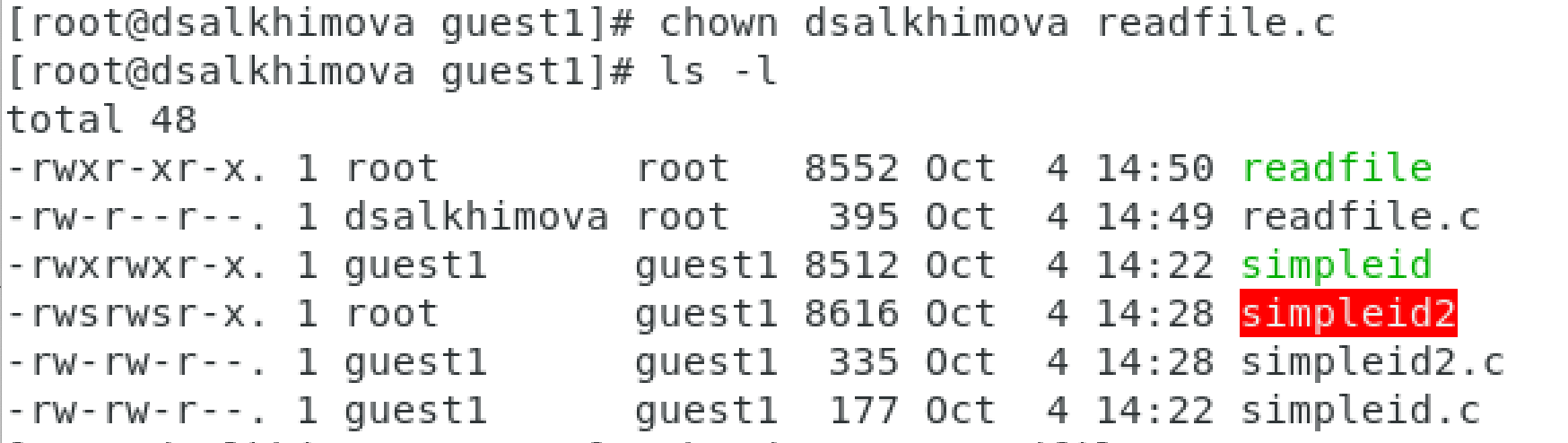
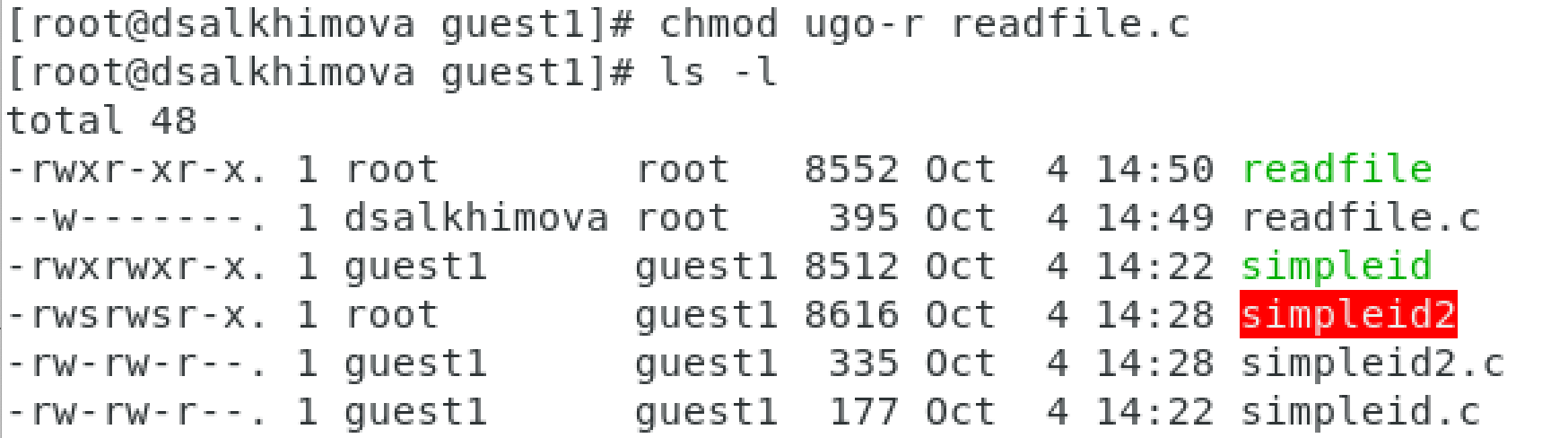
* 
* Сравнения вывода программ simpleid2 и id

Результаты вывода сходятся.

1. Проделала то же самое относительно SetGID-бита.([рис. 10](images5/10.png)).

* 
* Повтор операций для SetGID-бита

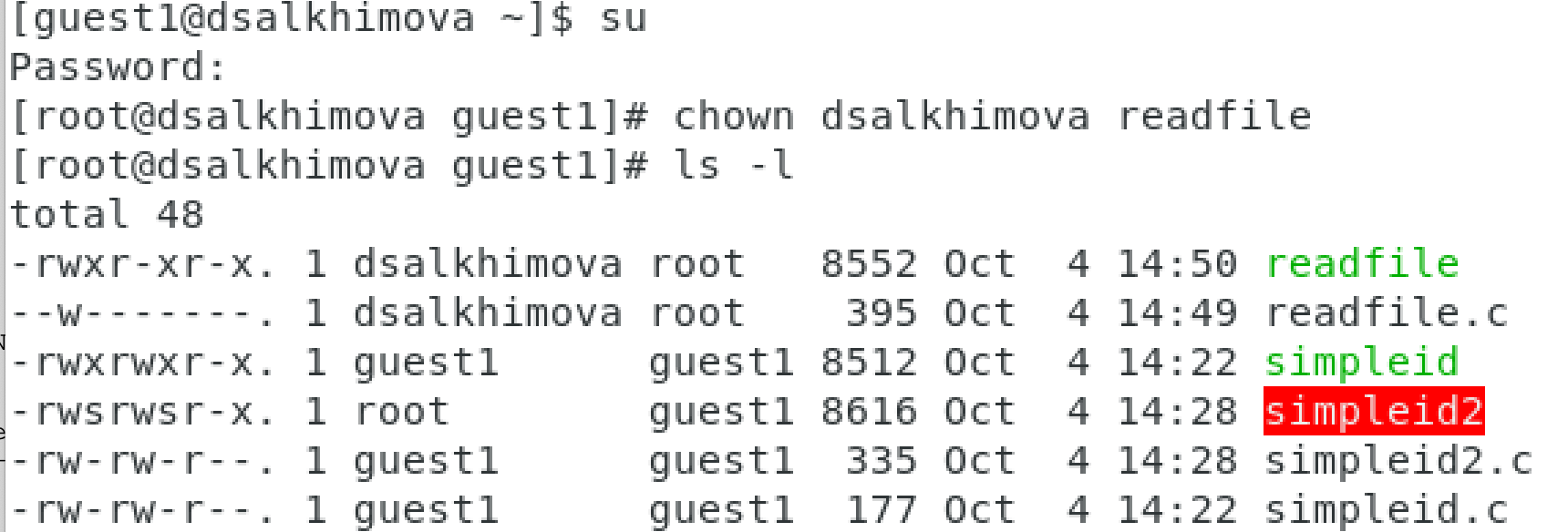
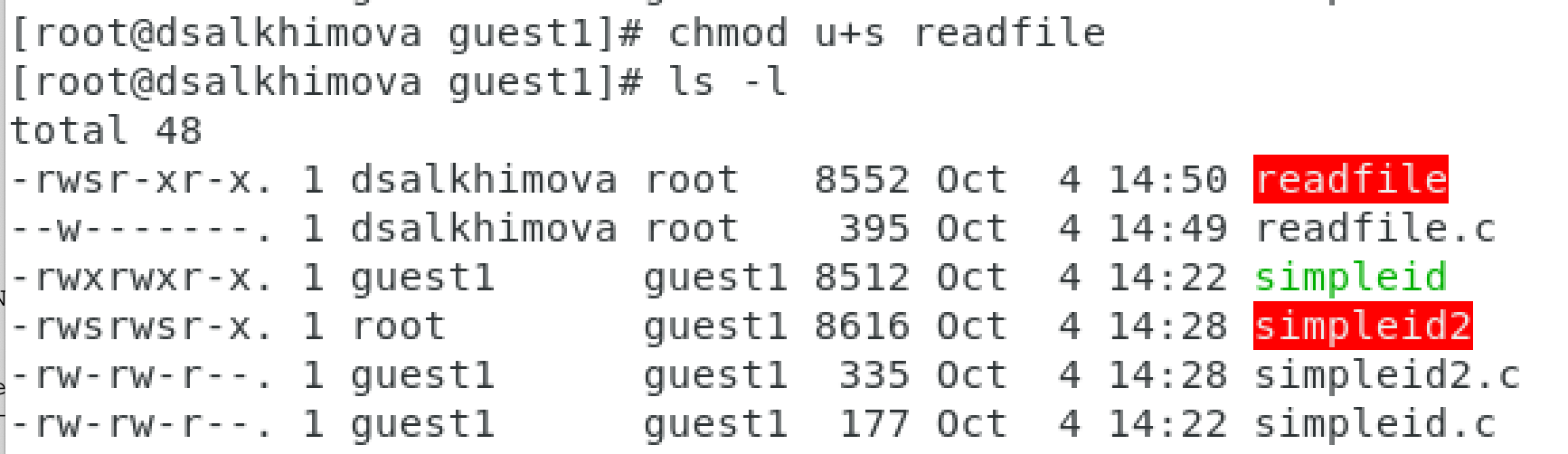
1. Создала и скомпилировала программу readfile.c ([рис. 11](images5/11.png)): #include <fcntl.h>  
   #include <stdio.h>  
   #include <sys/stat.h>  
   #include <sys/types.h>  
   #include <unistd.h>  
   int  
   main (int argc, char\* argv[])  
   {  
   unsigned char buffer[16]; size\_t bytes\_read;  
   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;  
   } 
2. Сменила владельца у файла readfile.c на dsalkhimova ([рис. 12](images5/12.png)) и изменила права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог ([рис. 13](images5/13.png)).

*   
  

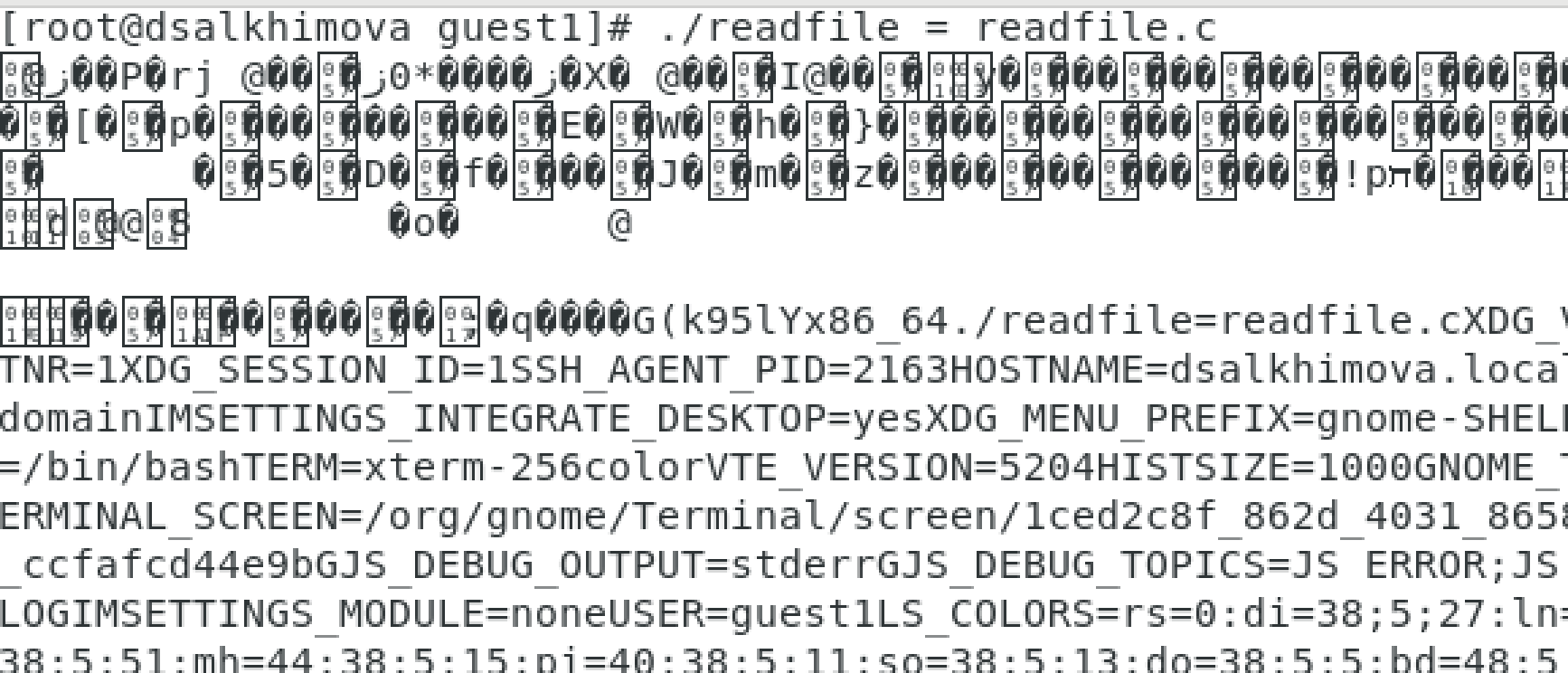
1. Проверила, что пользователь guest не может прочитать файл readfile.c ([рис. 14](images5/14.png)).

* 
* Проверка доступа guest к readfile

1. Сменила у программы readfile владельца ([рис. 15](images5/15.png)) и установила SetU’D-бит ([рис. 16](images5/16.png)).

*   
  

1. Проверила, что программа readfile может прочитать файл readfile.c ([рис. 17](images5/17.png)):

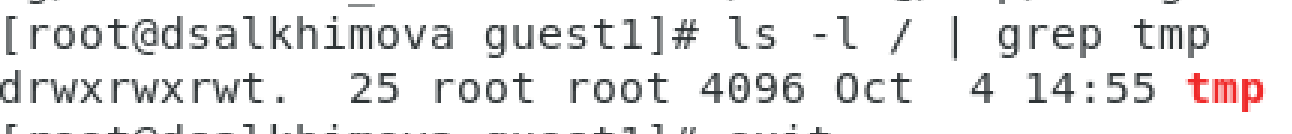
* 
* Чтение readfile readfile.c

1. Проверила, что программа readfile может прочитать файл /etc/shadow ([рис. 18](images5/18.png)):

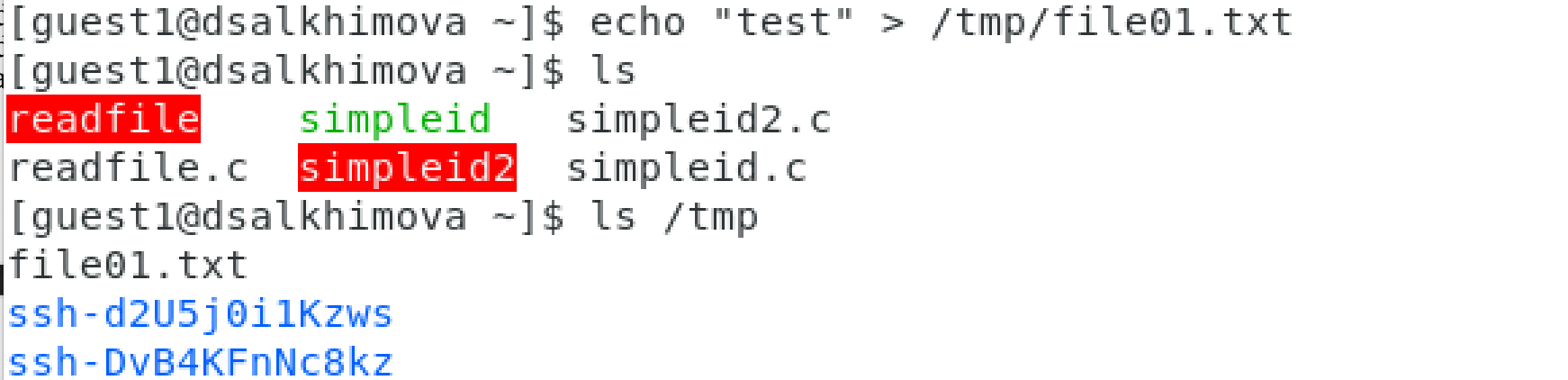
* 
* Чтение readfile /etc/shadow

## Часть 2

1. Выяснила, установлен ли атрибут Sticky на директории /tmp: ls -l / | grep tmp ([рис. 19](images5/19.png)). Атрибут t установлен.

* 
* Проверка атрибута Sticky

1. От имени пользователя guest создала файл file01.txt в директории /tmp со словом test: echo "test" > /tmp/file01.txt ([рис. 20](images5/20.png), [рис. 21](images5/21.png))

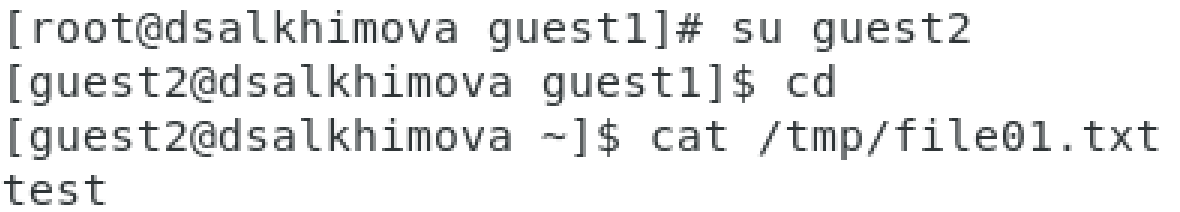
*   
  

1. Просмотрела атрибуты у только что созданного файла и разрешила чтение и запись для категории пользователей «все остальные» ([рис. 22](images5/22.png)):

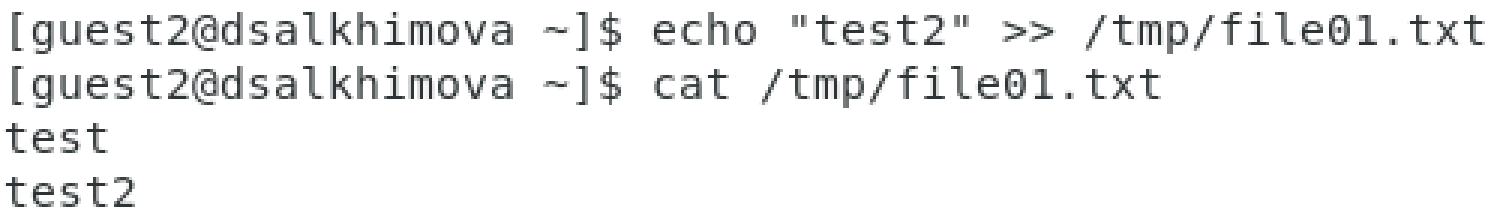
ls -l /tmp/file01.txt chmod o+rw /tmp/file01.txt ls -l /tmp/file01.txt

![Просмотр и изменение атрибутов file01.txt](images5/22.png){ #fig:022 width=70% }

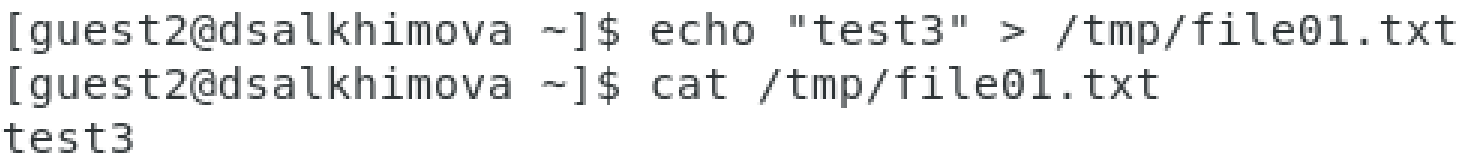
1. От пользователя guest2 (не являющегося владельцем) успешно прочитала файл /tmp/file01.txt: cat /tmp/file01.txt ([рис. 23](images5/23.png)):

* 
* Чение file01.txt пользователем guest2

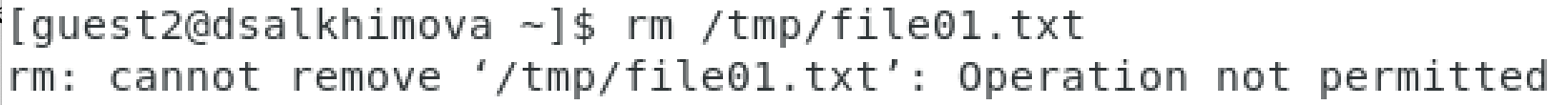
1. От пользователя guest2 успешно дозаписала в файл /tmp/file01.txt слово test2 командой echo "test2" >> /tmp/file01.txt и проверила содержимое файла командой cat /tmp/file01.txt ([рис. 24](images5/24.png)):

* 
* Дозапись file01.txt пользователем guest2

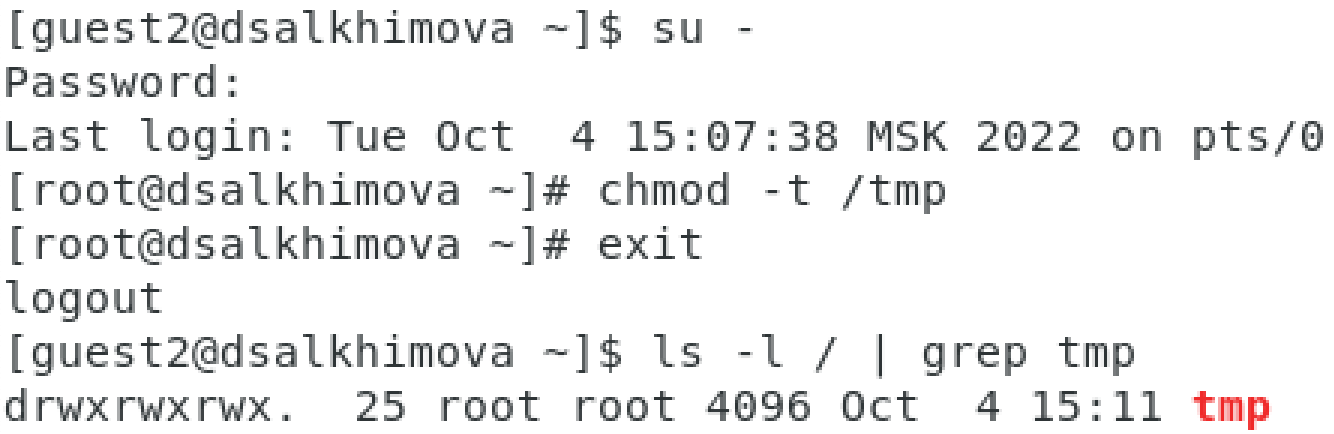
1. От пользователя guest2 успешно дозаписала в файл /tmp/file01.txt слово test3 командой echo "test3" > /tmp/file01.txt, стерев при этом все предыдущее содержимое файла, и проверила содержимое файла командой cat /tmp/file01.txt ([рис. 25](images5/25.png)):

* 
* Дозапись с удалением file01.txt пользователем guest2

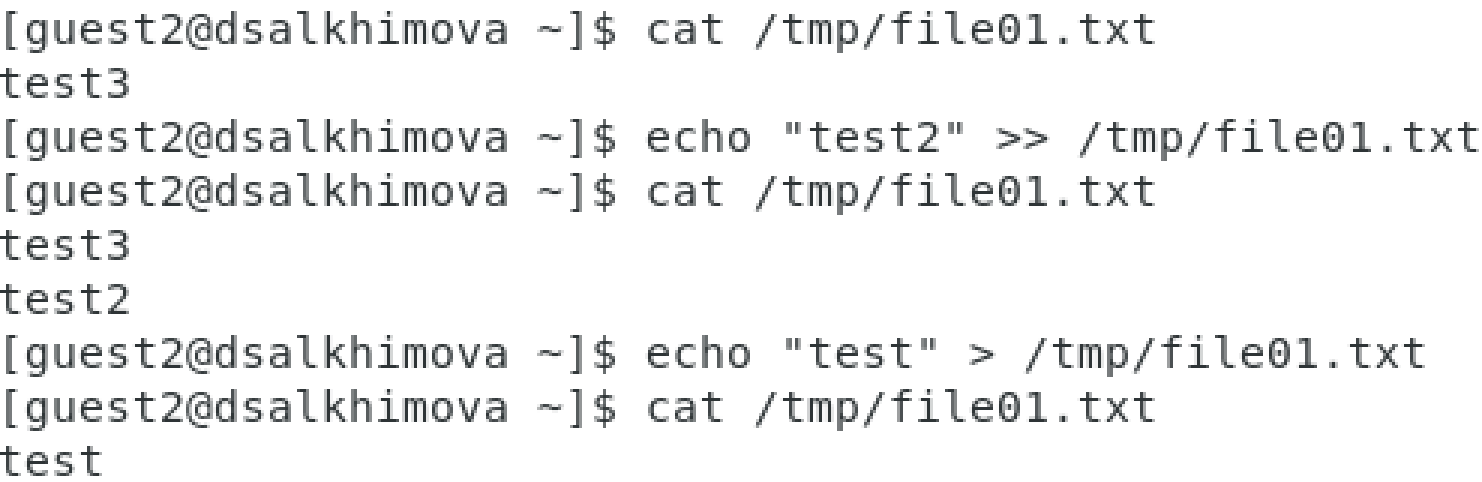
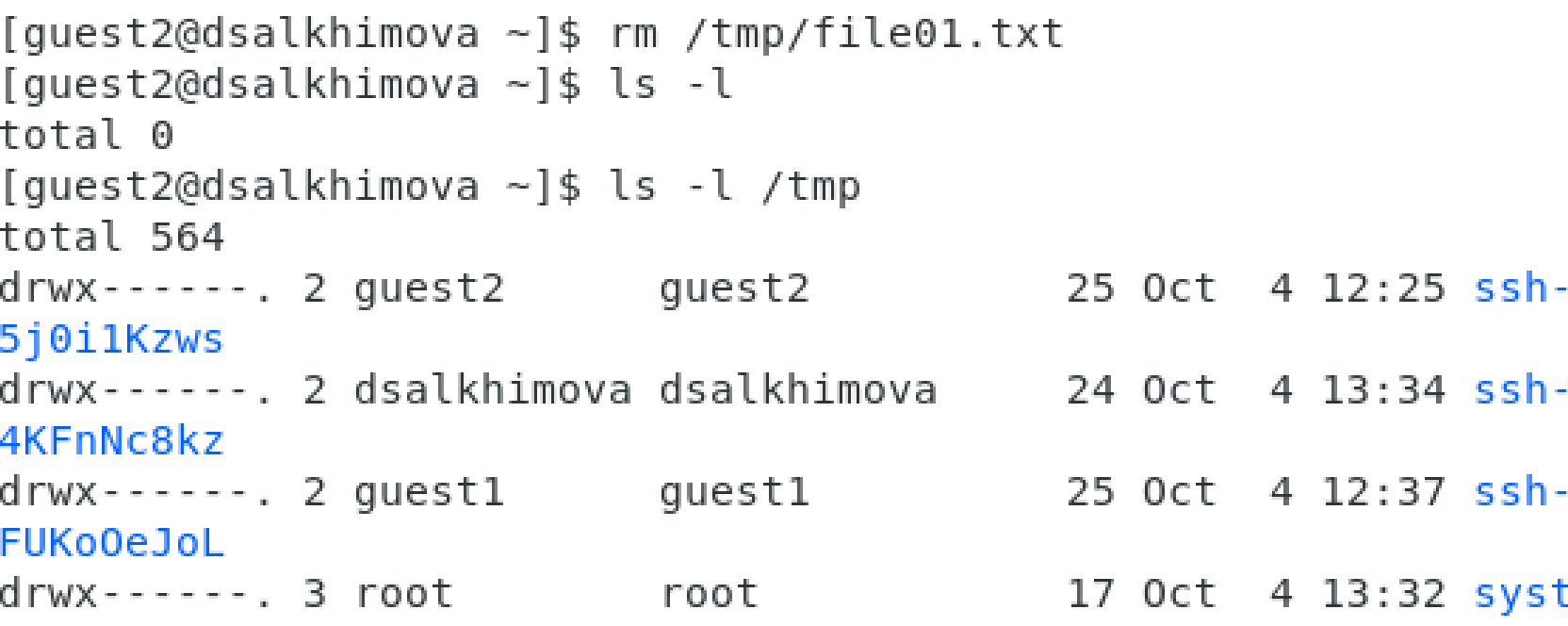
1. От пользователя guest2 попробовала удалить файл /tmp/file01.txt командой rm /tmp/fileOl.txt ([рис. 26](images5/26.png)). Операция запрещена.

* 
* Удаление file01.txt пользователем guest2

1. Повысила свои права до суперпользователя командой su - и выполнила после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp. Затем покинула режим суперпользователя командой exit. От пользователя guest2 проверила, что атрибута t у директории /tmp нет: ls -l / | grep tmp ([рис. 27](images5/27.png)):

* 
* Снятие атрибута Sticky

1. Повторила предыдущие шаги ([рис. 28](images5/28.png), [рис. 29](images5/29.png)). Изменений нет, кроме того, что теперь удалось удалить файл file01.txt от имени guest2.

*   
  

1. Повысила свои права до суперпользователя и вернула атрибут t на директорию /tmp ([рис. 30](images5/30.png)):

* 
* Возвращение атрибута Sticky на директрорию /tmp

# Выводы

В процессе выполнения данной лабораторной работы я научилась работать с механизмами изменения идентификаторов, применением SetUID- и Sticky-битов. Получила практических навыков 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

1. Описание лабораторной работы 5 - URL: <https://esystem.rudn.ru/pluginfile.php/1652171/mod_resource/content/2/005-lab_discret_sticky.pdf>
2. Команда SU в LINUX — URL: <https://losst.ru/komanda-su-v-linux?ysclid=l8u8lr4lj5312561951>
3. Команда SUDO в LINUX — URL: <https://losst.ru/komanda-sudo-v-linux?ysclid=l8uachqd98171389830>