Отчёт по лабораторной работе 5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Румянцева Александра Сергеевна

Содержание

[Цель работы 1](#_Toc87694360)

[Задание 1](#_Toc87694361)

[Теория 2](#_Toc87694362)

[Текстовый (символический) метод CHMOD 2](#_Toc87694363)

[Расширенные права доступа к файлам. 2](#_Toc87694364)

[Бит sticky для дирктории. 2](#_Toc87694365)

[Биты setgid и setuid для обычных файлов 3](#_Toc87694366)

[Выполнение лабораторной работы 3](#_Toc87694367)

[Подготовка к выполнению 3](#_Toc87694368)

[Выполнение основной частии лабораторной работы 3](#_Toc87694369)

[Исследование Sticky-бита 9](#_Toc87694370)

[Библиография 13](#_Toc87694371)

[Выводы 13](#_Toc87694372)

# 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.

# Задание

Лабораторная работа подразумевает изучение влияния дополнительных атрибутов на файлы пользователя и изучение механизмов изменения идентификаторов.

# Теория

## Текстовый (символический) метод CHMOD

Команда chmod позволяет вам изменять права доступа к файлу, используя либо символьный, либо числовой режим, либо ссылочный файл. Мы объясним режимы более подробно позже в этой статье. Команда может принимать один или несколько файлов и / или каталогов, разделенных пробелом, в качестве аргументов. Только root, владелец файла или пользователь с привилегиями sudo могут изменять права доступа к файлу. Будьте особенно осторожны при использовании chmod, особенно при рекурсивном изменении разрешений.

С численным методом мы знакомы с прошлых лаботаторных, изучим текстовый метод.

Синтаксис chmod команды при использовании символьного режима имеет следующий формат: chmod [OPTIONS] [ugoa…][-+=]perms…[,…] FILE…

Первый набор флагов ( [ugoa…]), флаги пользователей, определяет, для каких классов пользователей изменяются права доступа к файлу.

u - Владелец файла. g - Пользователи, которые являются членами группы. o - Все остальные пользователи. a- Все пользователи, идентичные ugo.

Второй набор флагов ( [-+=]), флагов операций, определяет, следует ли удалять, добавлять или устанавливать разрешения:

* Удаляет указанные разрешения.
* Добавляет указанные разрешения. = Изменяет текущие разрешения на указанные разрешения. Если после =символа не указано никаких разрешений , все разрешения из указанного пользовательского класса будут удалены.

Разрешения ( perms…) можно явно задать с помощью либо ноль , либо один или несколько из следующих букв: r, w, x, X, s, и t. Используйте одну букву из набора u, gи oпри копировании разрешений от одного к другому классу пользователей.

При настройке разрешений для нескольких пользовательских классов ( [,…]) используйте запятые (без пробелов) для разделения символьных режимов.

## Расширенные права доступа к файлам.

### Бит sticky для дирктории.

Можно установить бит sticky для директории с целью предотвращения удаления файлов пользователями, которые не являются их непосредственными владельцами. Бит sticky отображается в той же позиции, что и символ права исполнения x для пользователей, не являющихся владельцами директории и не состоящих в группе, владеющей директорией. Сам бит sticky отображается с помощью символа t (в том случае, если подразумевается наличие символа x) или символа T (в том случае, если символ хне должен выводиться ввиду отсутствия соответствующих прав доступа к директории).

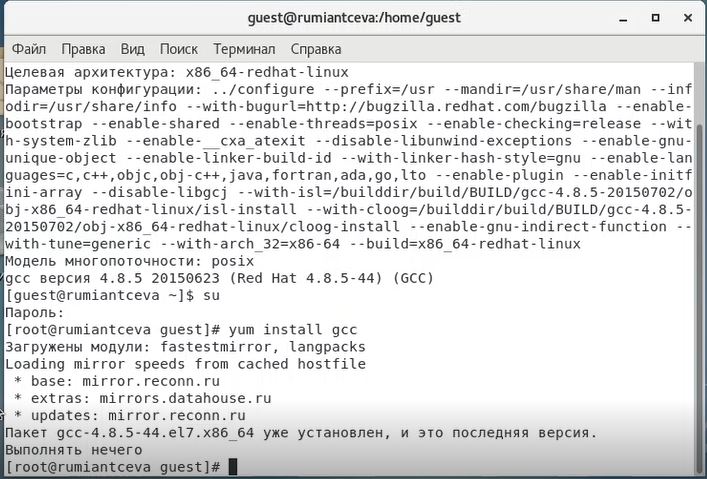
### Биты setgid и setuid для обычных файлов

Эти биты позволяют запускать исполняемый файл с правами пользователя, владеющего файлом, а не с правами пользователя, инициировавшего запуск файла. Это означает, что в том случае, если какой-либо пользователь запустит программу, принадлежащую пользователю root, причем для исполняемого файла программы будет установлен бит setud, то программа будет запущена от лица пользователя root. Такое поведение системы может оказаться опасным, но иногда оказывается и полезным для обхода ограничений безопасности.

# Выполнение лабораторной работы

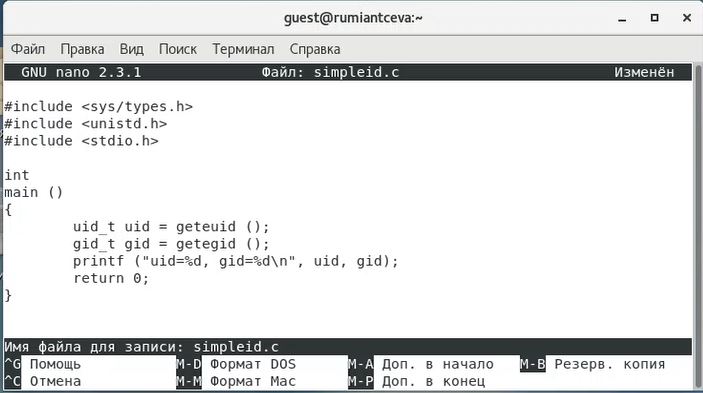
## Подготовка к выполнению

1. Проверила установлен ли компилятор gcc (рис. 1). В моём случаи он уже установлен.

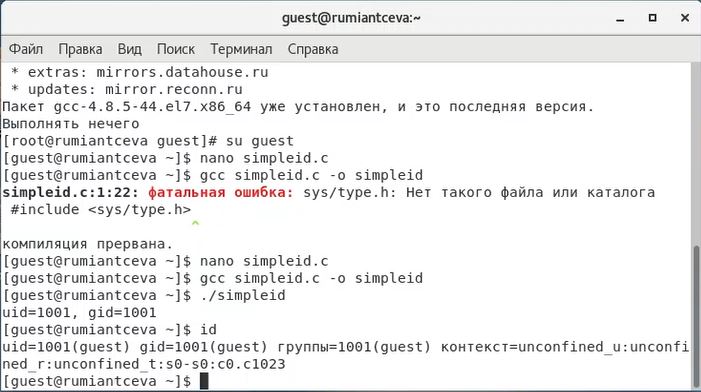
* 
* Figure 1: рис.1. Установка компилятора gcc.

## Выполнение основной частии лабораторной работы

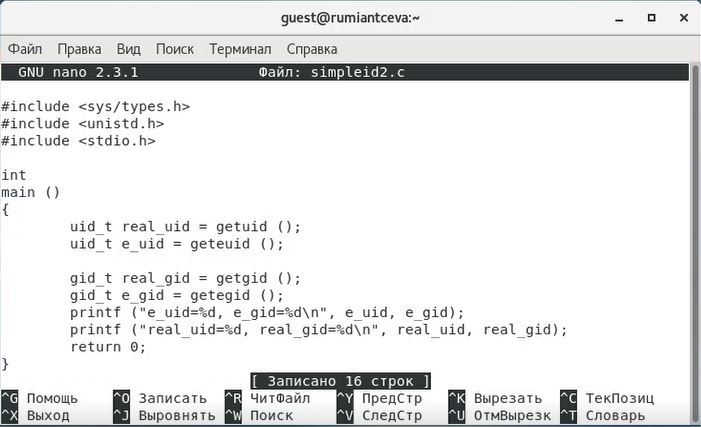
1. Вошла в систему от имени пользователя guest.
2. Создала программу simpleid.c (рис. 2).

* 
* Figure 2: рис.2. Программа simpleid.c

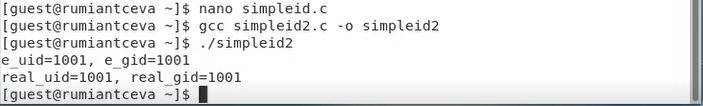
1. Скомплилировала программу и убедилась, что файл программы создан командой gcc simpleid.c -o simpleid (рис. 3).
2. Выполнила программу simpleid командой ./simpleid (рис. 3).
3. Выполнила системную программу id и сравнина полученный результат с данными предыдущего пункта задания (рис. 3). Видим, что пользователи и группы совпадают. При этом команда id вывела действительные идентификаторы, а программа вывела эффективные, но при этом они совпадают и выводят 1001, то есть пользователя guest.

* 
* Figure 3: рис.3. Компиляция программы simpleid, её выполнение и сравнение с командой id.

1. Усложнила программу, добавив вывод действительных идентификаторов, получившуюся программу назовала simpleid2.c (рис. 4).

* 
* Figure 4: рис.4. Программа simpleid2.c

1. Скомпилировала и запустила simpleid2.c командами gcc simpleid2.c -o simpleid2 и ./simpleid2 (рис. 5). Видим, что программа выводит эффективные и действительные идентификаторы пользователя и группы для файла. Видим, что везде это 1001, то есть пользователь guest.

* 
* Figure 5: рис.5. Компиляция программы simpleid2, её выполнение.

8-9. От имени суперпользователя выполнила команды:chown root:guest /home/guest/simpleid2 и chmod u+s /home/guest/simpleid2. Временно поменяв свои права с помощью su (рис. 6).

С помощью этих команд файлу simpleid2 изменила владельца и группу на root и guest соответственно (chown), а также установила на файл SetUID-бит (chmod).

1. Выполнила проверку правильности установки новых атрибутов и смены владельца файла simpleid2 командой ls -l simpleid2 (рис. 6).
2. Запустила simpleid2 и id командами ./simpleid2 и id (рис. 6). Сравнила результаты: действительные идентификаторы совпадают с выводом команды id - везде 0, то есть рут-пользователь. Так же важно заметить, что эффективные идентификаторы совпадают с действительными.

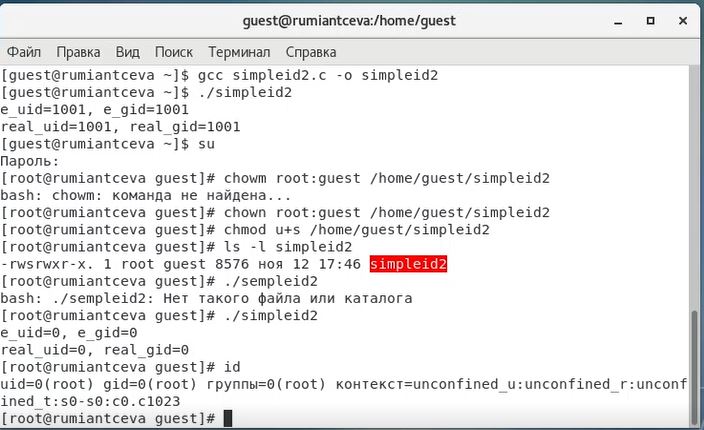


Figure 6: рис.6. Изменение владельца программы и установка SetUID-бита, проверка установки и изменения, запуск программы и команды id.

1. Проделала тоже самое относительно SetGID-бита (рис. 7)

Установка SetGID-бита отражается к команде ls, а сравнение выполнения программы и команды id дало следующие результаты: действительные идентификаторы совпадают с выводом команды id - везде 0, то есть рут-пользователь. Но так же важно заметить, что эффективные идентификаторы отличны от действительных: пользователь - 0, группа - 1001.



Figure 7: рис.7. Установка SetGID-бита, проверка установки, запуск программы и команды id.

1. Создала программу readfile.c (рис. 8).

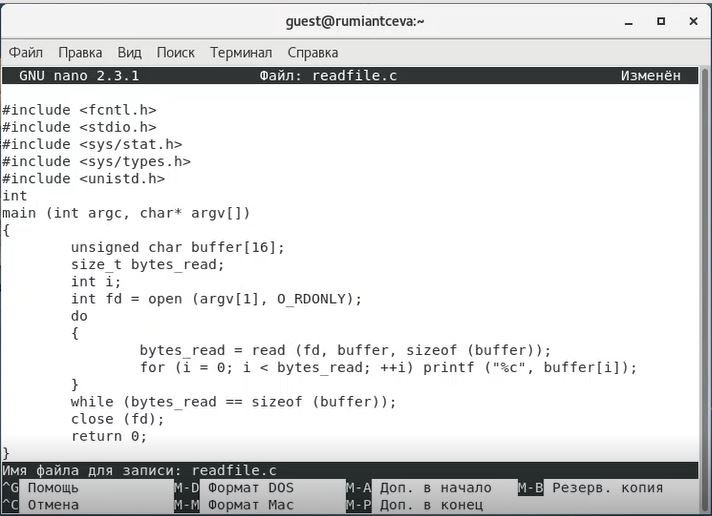


Figure 8: рис.8. Программа readfile.c

1. Откомпилировала её командой gcc readfile.c -o readfile (рис. 9).
2. Сменила владельца у файла readfile.c (chown) и изменила права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог. Использовала chmod a-r (рис. 9).
3. Проверила, что пользователь guest не может прочитать файл readfile.c командой cat (рис. 9).
4. Сменила у программы readfile владельца и установила SetUID-бит (рис. 9).

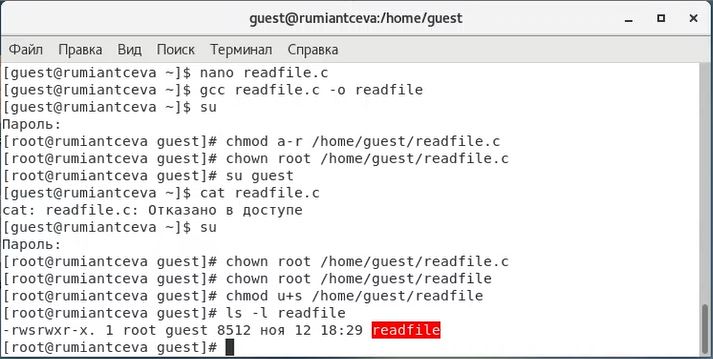


Figure 9: рис.9. Компиляция readfile и другие действия в соответствии с 14-17 пунктами.

1. Проверила, может ли программа readfile прочитать файл readfile.c. Да, может (рис. 10).

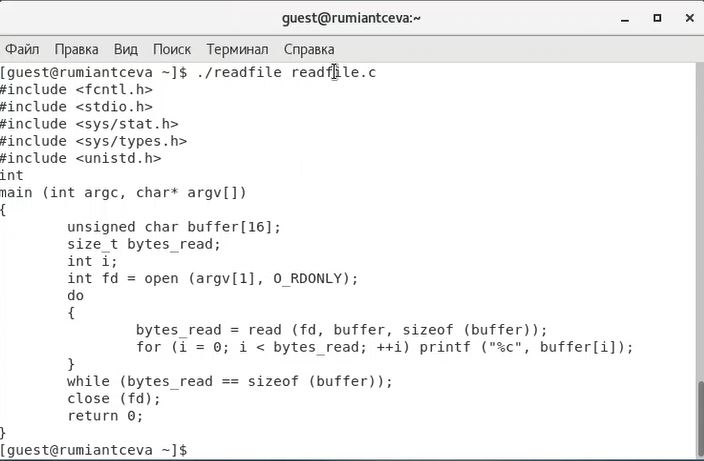


Figure 10: рис.10. Выполнение программы readfile с файлом readfile.c.

1. Проверила, может ли программа readfile прочитать файл /etc/shadow. Её выполненеи возможно в том числе, так как владельцем файла являктся root-пользователь (рис. 11).

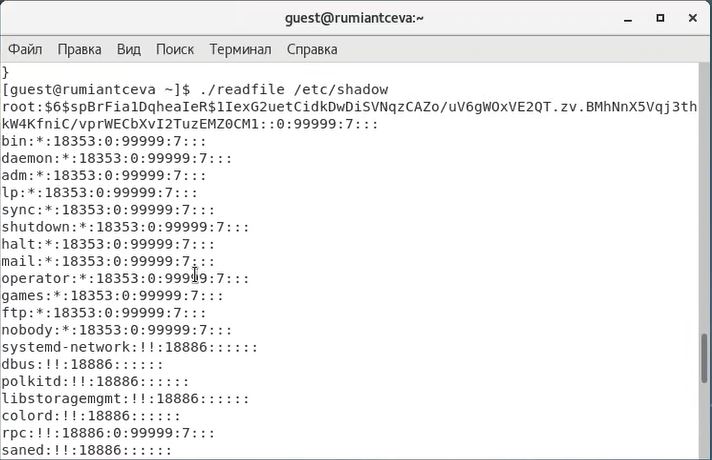
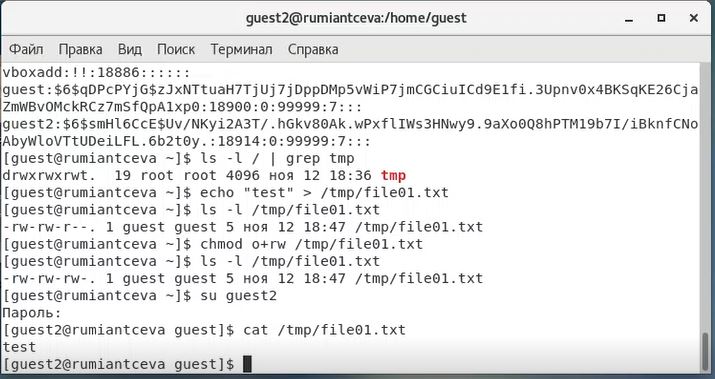


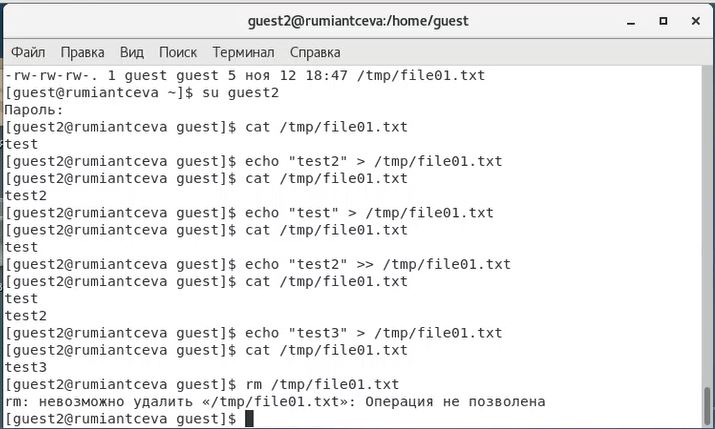
Figure 11: рис.11. Выполнение программы readfile с файлом /etc/shadow.

## Исследование Sticky-бита

1. Выяснила, установлен ли атрибут Sticky на директории /tmp, для чего выполнила команду ls -l / | grep tmp (рис. 12). Видим, что установлен, так как есть буква t.
2. От имени пользователя guest создала файл file01.txt в директории /tmp со словом test командой echo “test” > /tmp/file01.txt (рис. 12).
3. Просмотрела атрибуты у только что созданного файла и разрешила чтение и запись для категории пользователей «все остальные». Выполнила команды ls -l /tmp/file01.txt и chmod o+rw /tmp/file01.txt и ls -l /tmp/file01.txt (рис. 12).
4. От пользователя guest2 (не являющегося владельцем) попробовала прочитать файл /tmp/file01.txt командой cat /tmp/file01.txt (рис. 12).

* 
* Figure 12: рис.12. Выполнение пунктов 1-4 исследования Sticky-бита .

1. От пользователя guest2 попробовала дозаписать в файл /tmp/file01.txt слово test2 командой echo “test2” >> /tmp/file01.txt. Мне удалось выполнить операцию (рис. 13).
2. Проверила содержимое файла командой cat /tmp/file01.txt (рис. 13).
3. От пользователя guest2 попробовала записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой echo “test3” > /tmp/file01.txt. Мне удалось выполнить операцию (рис. 13).
4. Проверила содержимое файла командой cat /tmp/file01.txt (рис. 13).
5. От пользователя guest2 попробовала удалить файл /tmp/file01.txt командой rm /tmp/fileOl.txt. Мне нее удалось удалить файл (рис. 13).

* 
* Figure 13: рис.13. Выполнение пунктов 5-9 исследования Sticky-бита .

Можем сделать вывод, что разрешена дозапись в файл, запись в файл, но мы не можем удалять файл из директории, на которую установлен атрибут Sticky.

1. Повысила свои права до суперпользователя следующей командой su - и выполнила после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp (рис. 14).
2. Покинула режим суперпользователя командой exit (рис. 14).
3. От пользователя guest2 проверила, что атрибута t у директории /tmp нет: ls -l / | grep tmp (рис. 14).
4. Повторила предыдущие шаги (рис. 14). Видим, что дозапись и запись так же разрешены, но при этом удалось и удалить файл.

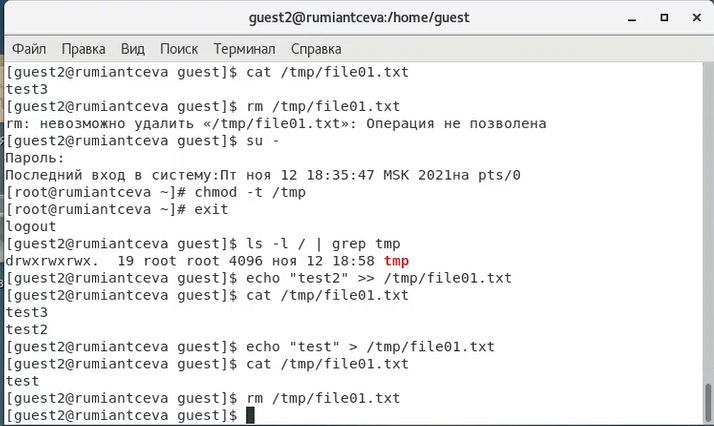


Figure 14: рис.14. Выполнение пунктов 10-13 исследования Sticky-бита .

1. Мне удалось удалить файл от имени пользователя, не являющегося его владельцем. Именно об этом и была часть теории выше.
2. Повысила свои права до суперпользователя и верните атрибут t на директорию /tmp: su -, chmod +t /tmp, exit (рис. 15).

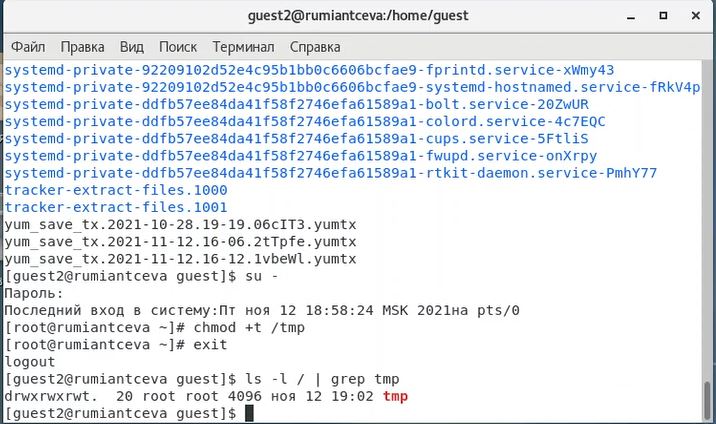


Figure 15: рис.15. Возврашение атрибута t на директорию.

# Библиография

1. ТУИС РУДН
2. Статья “КОМАНДА CHMOD В LINUX (ПРАВА ДОСТУПА К ФАЙЛАМ)” на сайте baks.ru <https://baks.dev/article/terminal/chmod-command-in-linux-file-permissions>
3. Статья “Фундаментальные основы Linux” на сайте zalinux.ru <http://rus-linux.net/MyLDP/BOOKS/Linux_Foundations/31/ch31.html>

# Выводы

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.