Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Отчет по дисциплине   
«Операционные системы»

Лабораторная работа №3  
«Управление правами пользователей»

Студенты: Клоченко И.Е., Юсупов Д.А.

Преподаватель: Коротяев А.Н.

Группа: РИ-300024

Екатеринбург

2023

**Оглавление**

[Задание 1 3](#_Toc132287374)

[1. Создание директории *test* и файла *tetst* 3](#_Toc132287375)

[2. Изменения прав доступа к файлу 4](#_Toc132287376)

[3. Добавление прав доступа к файлу 5](#_Toc132287377)

[4. Изменение владельца файла 5](#_Toc132287378)

[5. Изменение принадлежности файла к группе 6](#_Toc132287379)

[6. Установление права на запись файла для всех пользователей кроме владельца 6](#_Toc132287380)

[7. Задание прав доступа в форме битовой маски 7](#_Toc132287381)

[8. Создание жесткой и символической ссылки на файл 7](#_Toc132287382)

[9. Использование *ACL* для пунктов 2-3,6,7 9](#_Toc132287383)

[Задание 2 11](#_Toc132287384)

[1. Создание нескольких пользователей 11](#_Toc132287385)

[2. Выполнение команд от лица первого пользователя 11](#_Toc132287386)

[3. Создание файлов от лица второго пользователя 11](#_Toc132287387)

[4. Выполнение каскада команд для каждого каталога (*cat* и *ls -la*) 12](#_Toc132287388)

[Задание 3 13](#_Toc132287389)

[1. Определение текущей маски пользователя 13](#_Toc132287390)

[2. Задание маски 14](#_Toc132287391)

[3. Попытка входа в созданную папку 14](#_Toc132287392)

[4. Права у созданных папки и файла 15](#_Toc132287393)

[5. Попытка с маской 000 15](#_Toc132287394)

[6. Принцип работы команды *umask* 15](#_Toc132287395)

[Задание 4 16](#_Toc132287396)

[Вывод 20](#_Toc132287397)

**Цель работы:**

знакомство с распределением прав доступа к файлам и директориям.

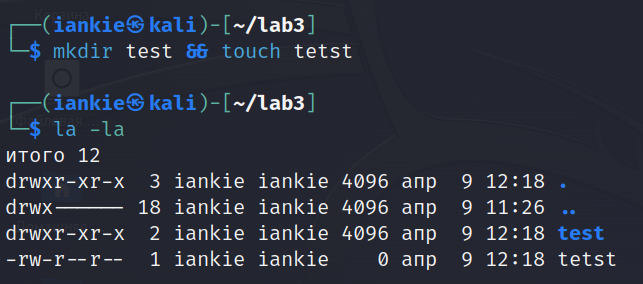
**Ход работы:**

Комментарий: Лабораторная работа выполнена в ОС Kali Linux, установленной на виртуальной машине VMware Workstation. Выбор этого дистрибутива обусловлен опытом взаимодействия с этим дистрибутивом и удобством использования и выполнения лабораторной работы по сравнению с CentOS 7. Пользователем по умолчанию установлен пользователь с логином *iankie*, имеющий права суперпользователя.

# Задание 1

1. Создание директории *test* и файла *tetst*

Создадим указанные директорию и файл:



**Рисунок 1.** Атрибуты доступа директории test и файла tetst

Вывод команды *ls -la* (в Kali Linux можно использовать короткую версию – *la*) представляет возможность исследовать атрибуты доступа к созданным директории и файлу.

Расшифровка атрибутов:

У директорий первым атрибутом (тип объекта) идет – *d* (*directory*), у файлов этот атрибут отсутствует (прочерк «-»).

Таблица .Атрибуты прав доступа файла и директории

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Группа | | | |
| Файл/директория | Владелец (*owner*) | Группа владельца (*group*) | Другие (*others*) |
| *test* | *rwx* | *rx* | *rx* |
| *tetst* | *rw* | *r* | *r* |

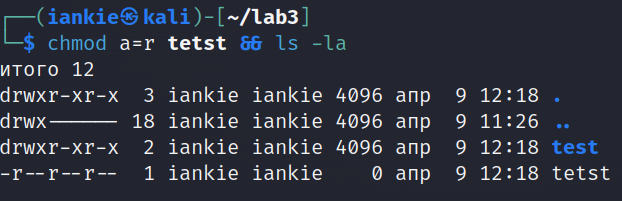
*r* – чтение,

*w* – запись,

*x* – исполнение.

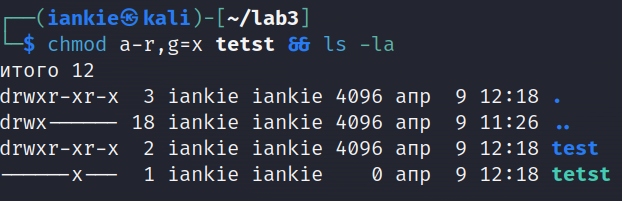
1. Изменения прав доступа к файлу

Изменим права доступа к файлу на только чтение для всех пользователей:



**Рисунок 2.** Изменение прав доступа к файлу (только чтение для всех пользователей)

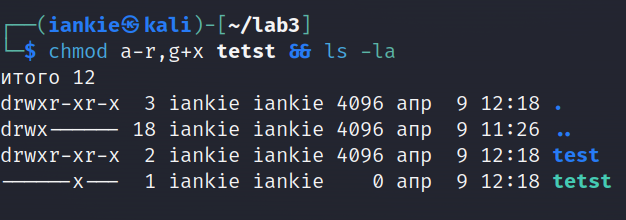
Теперь изменим права доступа к файлу на только исполнение и только для владельца группы (не совсем ясна трактовка – для владельца файла или для группы, владельцем которой является файл, в примере ниже выполнено второе):



**Рисунок 3**. Изменение прав доступа к файлу (только исполнение и только для группы, которая владеет файлом *tetst*)

1. Добавление прав доступа к файлу

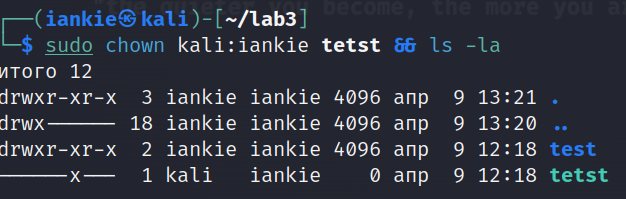
Добавим права доступа к файлу на исполнение для владельца группы. Процедура изменения и добавления прав отличается только знаком («=» для изменения прав, «+» - для добавления):



**Рисунок 4**. Добавление права доступа к файлу (исполнение для группы, владеющей файлом *tetst*)

1. Изменение владельца файла

Изменим владельца файла на *kali*:



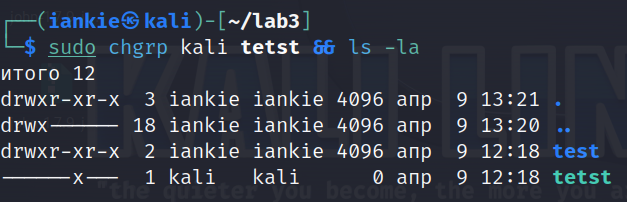
**Рисунок 5**. Изменение владельца файла

Важно учесть, что смена владельца, также как и смена группы, владеющей файлом, выполняется только с правами суперпользователя.

1. Изменение принадлежности файла к группе

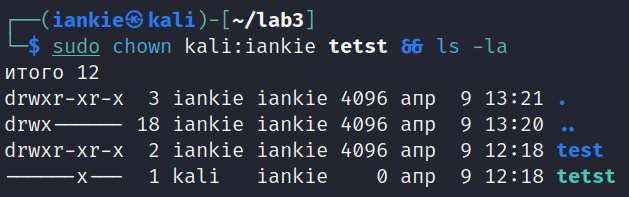
Сменить группу можно двумя способами: через команду *chown* или через команду *chgrp.*

Сменим группу, владеющую файлом (на *kali*):



**Рисунок 6**. Смена группы через команду *chgrp*

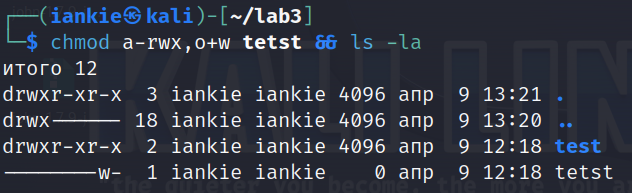
Сменим группу, владеющую файлом (на *iankie*):



**Рисунок 7**. Смена группы через команду *chown*

1. Установление права на запись файла для всех пользователей кроме владельца

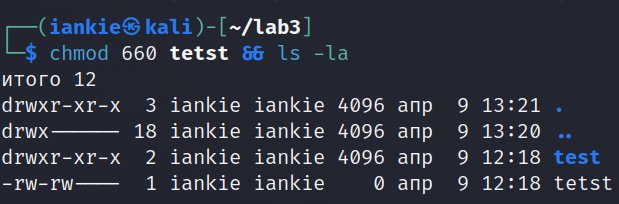
Установим право доступа к файлу на запись файла для всех пользователей кроме владельца:



**Рисунок 8**. Установление права на запись для остальных пользователей, но не для владельца

1. Задание прав доступа в форме битовой маски

Установим права чтения и редактирования для владельца и группы владельца и лишим всех прав всех остальных пользователей с помощью одной команды в форме битовой маски:

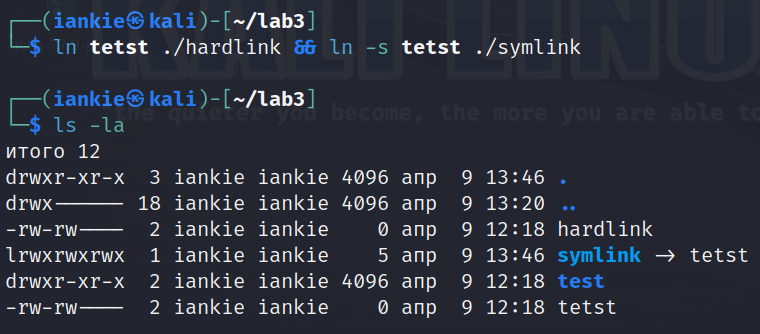


**Рисунок 9**. Установка прав на файл с помощью битовой маски

Права на чтение (4 бита) и запись (2 бита) – 6 бит. Эти права необходимо дать владельцу и группе владельца – 66. Остальные пользователи не должны иметь никаких прав на файл, поэтому 0 бит. Итого: 660 – битовая маска для задания прав на файл по условию, указанному выше.

1. Создание жесткой и символической ссылки на файл

Создадим жесткую (*ln filename*) и символическую (*ln -s filename*) ссылки на файл:

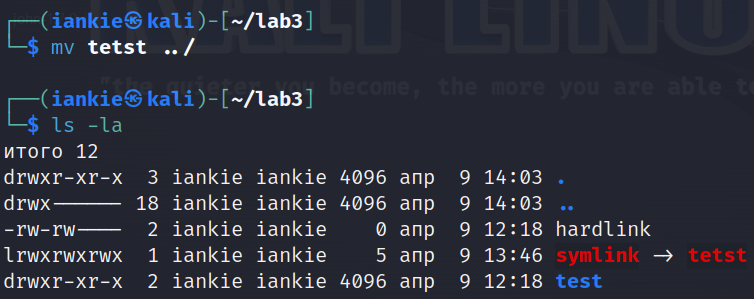


**Рисунок 10**. Создание жесткой и символической ссылок на файл

Жесткая ссылка имеет те же права, что и исходный файл, потому что жесткая ссылка ссылается на ту же структуру. Если изменить права доступа у исходного файла, права у жесткой ссылки также изменятся.

Символическая ссылка отличается тем, что:

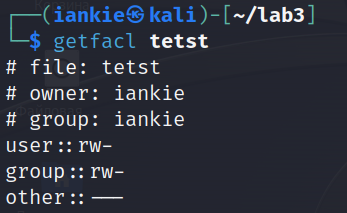
* + 1. В результате команды *ls -la* показывает на что происходит ссылка
  1. Имеет атрибут типа *l* (*link*)
  2. Имеет свои собственные права доступа и не зависит от изменения прав на исходный файл. Ссылка может существовать даже если исходный файл удален или перемещен:



**Рисунок 11**. Пример вывода *ls -la* (файл *tetst* перемещен в другую директорию)

1. Использование *ACL* для пунктов 2-3,6,7

Узнать, какие есть атрибуты доступа прав к файлу можно с помощью команды *getfacl*:

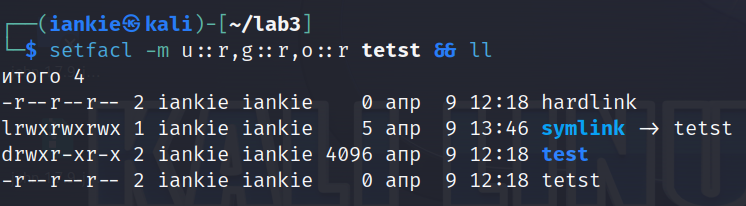


**Рисунок 12**. Просмотр прав доступа к файлу с помощью команды *getfacl*

Изменить права доступа к файлу можно через команду *setfacl*:

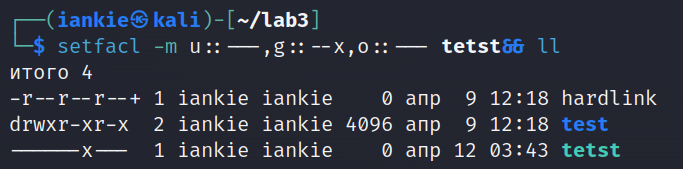
*setfacl -m u:username:[rwx],g:groupname:[rwx],o:username:[rwx]*

Изменим права доступа к файлу на только чтение для всех:



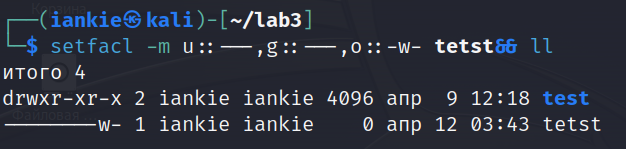
**Рисунок 13**. Права доступа к файлу на чтение для всех (*getfacl*)

Изменим права доступа к файлу на только исполнение и только для владельца группы (группа, владелец которой – владелец файла):



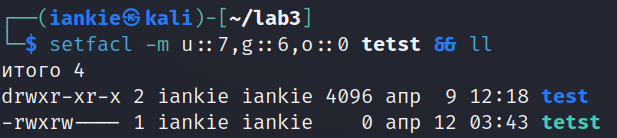
**Рисунок 14**. Изменение прав на только исполнение и только для группы-владельца файла (*setfacl*)

Установим право на запись файла для всех пользователей кроме владельца:



**Рисунок 15**. Установление права на запись для всех пользователей кроме владельца (*setfacl*)

Зададим права доступа (любые) для владельца, чтение и редактирование для группы и никаких прав для остальных пользователей с помощью битовой маски:

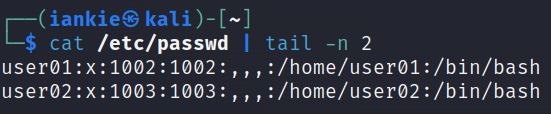


**Рисунок 16**. Задание прав в форме битовой маски (*setfacl*)

# Задание 2

1. Создание нескольких пользователей

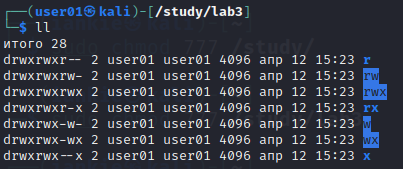
Создадим несколько пользователей:



**Рисунок 17**. Создано 2 пользователя *user01* и *user02*

1. Выполнение команд от лица первого пользователя

Выполним указанные в методических материалах команды и проверим, соответствие прав доступа файлов и каталогов заданным:



**Рисунок 18**. Каталоги с заданными правами доступа

Некоторые из установленных прав доступа лишены смысла, например *r* или *w* (в одиночном использовании), потому что не дают как таковых прав, им в пару нужен атрибут x для открытия директории.

1. Создание файлов от лица второго пользователя

Если попытаться от лица второго пользователя создать файл в каждой директории, созданной первым пользователем, то будет выведена ошибка «Отказано в доступе»:



**Рисунок 19**. Ошибка «Отказано в доступе»

Исключением будут те директории, в которых установлены права «*rwx*» и «*wx*».

Если попробовать провести те же действия, результат будет положительным для каждого каталога, так как ранее мы установили все права для владельца и участников одноименной группы пользователя.

Тот же результат получен при попытке выполнить действия от лица суперпользователя. Причина – есть права доступа ко всему (особенность привилегий суперпользователя).

Итоговая таблица:

Таблица 2. Таблица результативности попыток создания файлов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *x* | *w* | *wx* | *r* | *rx* | *rw* | *rwx* |
| *root* | *+* | *+* | *+* | *+* | *+* | *+* | *+* |
| *User01* | *+* | *+* | *+* | *+* | *+* | *+* | *+* |
| *User02* | *-* | *-* | *+* | *-* | *-* | *-* | *+* |

1. Выполнение каскада команд для каждого каталога (*cat* и *ls -la*)

Выполнив последовательно команды *ls -la* и *cat* для созданных каталогов и файлов соответственно, успешно выполнится команды, обращающиеся каталогам/файлам:

* *ls -la*: *rx* и *rwx* (обращение к *r и rw* вернут только содержание и напомнят о «Отказано в доступе», скрыв детали прав доступа и прочую информацию, выводимую командой; к остальным каталогам доступа нет никакого)
* *cat*: *x*, *wx*, *rx*, *rwx* (попытка чтения *r*, *w*, *rw* приведет к ошибке «Отказано в доступе»)

Такое поведение программ объясняется следующими особенностями атрибутов прав доступа:

* Атрибут *r* позволяет просмотреть содержимое каталога, но без остальных атрибутов (дата, владелец, группа и тд). Успешно работает с атрибутом *x* (*rx*, *rwx*) на просмотр каталога.
* Атрибут *w* здесь не важен для рассмотрения.
* Атрибут *x*. Для каталогов значит разрешить переместиться в них (поэтому успешно работает команда *ls -la* для каталогов с атрибутом *x*). При условии, что у файла присутствует атрибут *r* (все созданные файлы по умолчанию имеют атрибуты 644), атрибут *x* у каталога позволяет просмотреть этот файл, при этом у каталога может быть только этот атрибут.

Резюмируем все выше сказанное: просмотреть содержимое каталогов можно при присутствии у них атрибутов *r* и *x*, прочитать файлы внутри каталогов можно, если у этих каталогов установлен минимум атрибут *x,* а сам файл имеет минимум атрибут *r*.

# Задание 3

1. Определение текущей маски пользователя

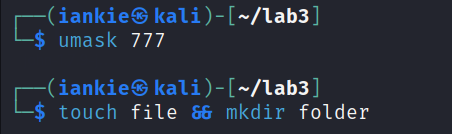
Определим текущую маску пользователя:



**Рисунок 20**. Текущая маска пользователя

1. Задание маски

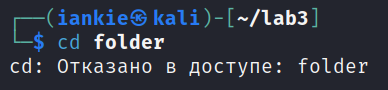
Зададим *umask* 777, создадим файл и отдельно папку:



**Рисунок 21**. Задание маски 777, создание файла и папки

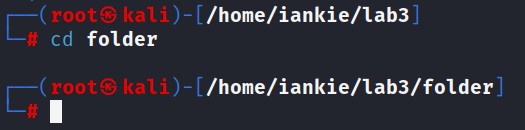
1. Попытка входа в созданную папку

Попытаемся войти в созданную папку, получаем ошибку «Отказано в доступе»:



**Рисунок 22**. Ошибка «Отказано в доступе»

Попробуем зайти от лица суперпользователя:

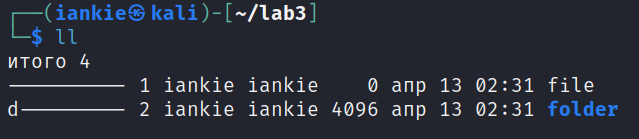


**Рисунок 23**. Попытка входа от лица суперпользователя

Успешно.

1. Права у созданных папки и файла

Посмотрим атрибуты прав доступа у созданных файла и папки:

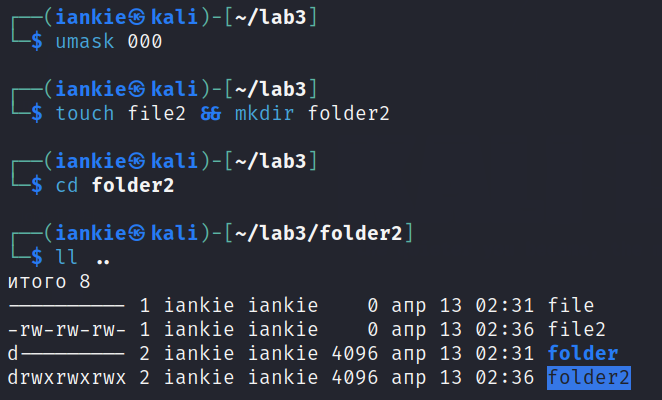


**Рисунок 24**. Атрибуты прав доступа файла и папки

Как видно, никаких.

1. Попытка с маской 000

Попробуем повторить все выше пройденное с маской 000:



**Рисунок 25**. Маска 000

Результат: все шаги были успешно выполнены, для текущего пользователя установлены все возможные права на файл *file2* (кроме прав на исполнение) и папку *folder2*.

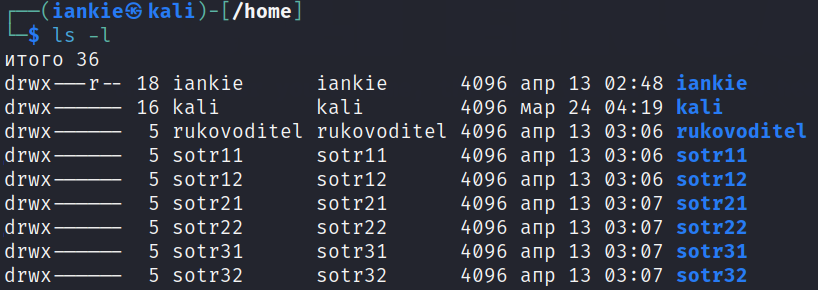
1. Принцип работы команды *umask*

Команда *umask* устанавливает для текущего пользователя режим создания файлов (директорий) маской. Число, подаваемое в качестве аргумента команде *umask* – это инвертированное число битовой маски атрибутов прав доступа. Вот несколько примеров соответствия:

* *umask* 777 = *chmod* 000
* *umask* 776 = *chmod* 001
* *umask* 022 = *chmod* 755 и т.д.

# Задание 4

Создаем сотрудников *sotr11, sotr12, sotr21, …, rukovoditel*.



**Рисунок 26**. Создание сотрудников

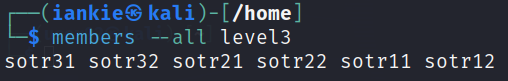
Для разграничения по уровням доступа было принято решение создать 3 группы для каждого уровня информации: *level1, level2, level3*.

Сотрудников 3 уровня доступа, добавляем группу *level3*.

Сотрудников 2 уровня доступа добавляем в группы *level2* и *level3*.

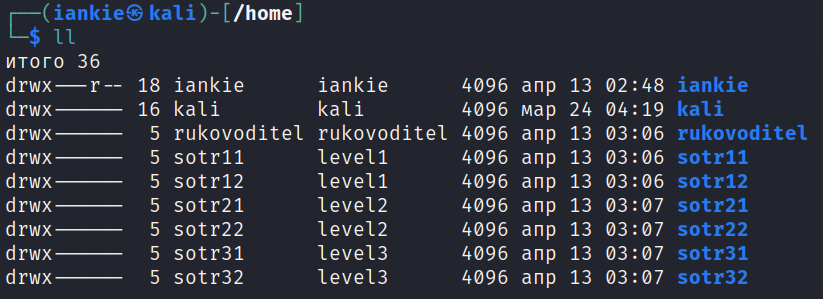
Сотрудников 1 уровня доступа и руководителя добавляем в группы *level1, level2 и level3*.

Пример содержания группы *level3*:



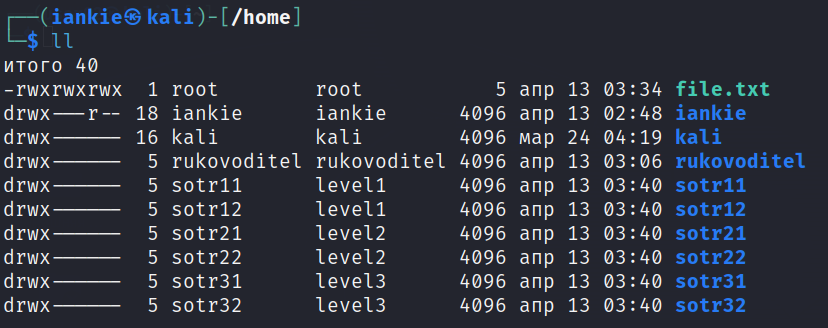
**Рисунок 27**. Содержание группы *level3*

Далее присваиваем домашним каталогам сотрудников в качестве владельцев-групп level1, level2, level3:



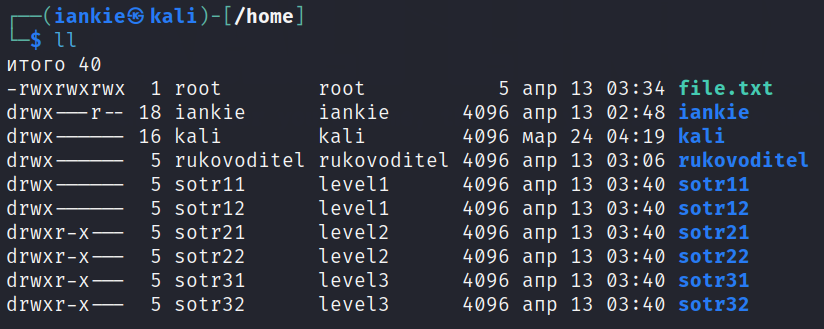
**Рисунок 28**. Смена владельцев-групп

Устанавливаем права группы на каждый домашний каталог сотрудника:



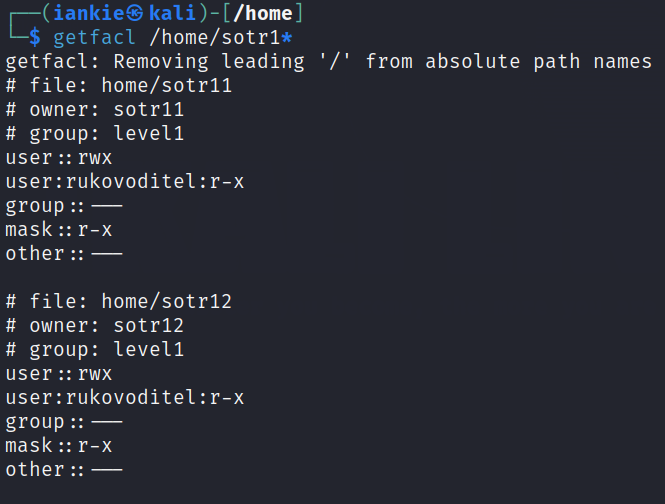
**Рисунок 29**. Установка групп-владельцев для домашних директорий сотрудников

Далее установим права группы на домашние директории:



**Рисунок 30**. Установка прав группы на домашние директории сотрудников

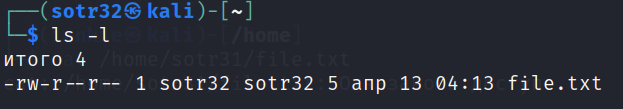
Так как у группы *level1* нет прав для просмотра домашних директорий сотрудников 1 уровня доступа (а руководитель добавлен в эту группу), нужно добавить персональные права для руководителя:



**Рисунок 31**. Задание персональных прав на домашние директории сотрудников с 1 уровне доступа руководителю

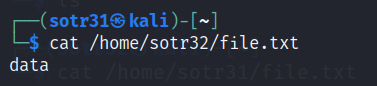
Возможность настройки привилегий таким образом лишает смысла добавления руководителя в группы уровней доступа, поэтому можно убрать этого пользователя из списков групп уровней доступа и назначить персональные права на просмотр для домашней директории каждого сотрудника, но, в силу упрощенной реализации, для 2 и 3 уровней оставим настоящую организацию привилегий.

Проверим корректность работы настроенной привилегированной системы. У сотрудника sotr32 в домашней директории есть файл:



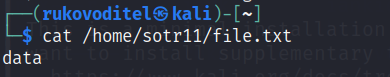
**Рисунок 32**. Файл *sotr32*

Попробуем прочесть этот файл от лица пользователя *sotr31*:



**Рисунок 33**. Чтение файла *sotr32*

Успешно. Теперь проверим то же самое для чтения руководителем файла *sotr11*:



**Рисунок 34**. Чтение файла *sotr11* руководителем

Успех. Система настроена корректно.

# Вывод

В ходе лабораторной работы были освоены навыки администрирования учетных записей пользователей, а именно работа с правами доступа к файлам и директориям. На практике доведено до совершенства владение такими утилитами, как *chmod*, *chown*, *chgrp*, *setfacl*, *umask*.

В выводе хотелось бы упомянуть про сравнение утилит *chmod* и *setfacl*. На первый взгляд, выполняют эти утилиты одну и ту же функцию – распределение прав доступа к файлам и директориям, но утилита *setfacl* имеет более расширенный арсенал возможностей, например, установка персональных прав для пользователя или группы (эта возможность помогла выполнить задание 4). Но использовать ее для простых задач, где утилита *chmod* справится с более короткой и простой версией написания команд, не рекомендуется.

Выполнено комплексное задание по созданию и настройке привилегированной мини-системы пользователей. Тем самым закреплен навык работы с функционалом утилит, упомянутых выше.