7.Markovskij I. N., Poznjakov S. I., Men'shikov V. I. Funkcional'naja gotovnost' «chelovecheskogo jelementa» pri vosprijatii navigacionnoj informacii ot jekspertnyh sistem // Rybnoe hozjajstvo. 2013. № 6. P. 93 – 95.

8.Osnovnye processy v strukturah bezopasnoj jekspluatacii sudna : monografija / F. D. Kukui, A. N. Anisimov, A. A. Anisimov ; pod obshh. red. V. I. Men'shikova ; Feder. agentstvo po rybolovstvu, FGOU VPO «Murm. gos. tehn. un-t». Murmansk : Izd-vo MGTU, 2008. 184 p. : il.

9.Pomaho S. V., Smirnov A. Zh., Men'shikov V. I. Ispol'zovanie associativno-strukturnogo podhoda pri opisanii avarijnoj situacii i variantov ee razvitija // Vestn. MGTU: Trudy Murman. gos. tehn. un-ta. 2011. T.14. № 4. P. 737 – 739.

10.Koks D., Smit V. Teorija vosstanovlenija. M. : Sovetskoe radio, 1967. 165 p.

ПРОГРАММА ОБЕСПЕЧЕНИЯ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ РЕЖИМОМ ДОСТУПА К РЕСУРСАМ ФАЙЛОВОЙ СИСТЕМЫ

Кварацхелия Леонид Дмитриевич
Студент СарФТИ НИЯУ МИФИ
факультет Информационных технологий и электроники
г.Саров, Россия

АННОТАЦИЯ

Статья посвящена файловым системам, правам доступа к её ресурсам и разработке приложения для автоматизации получения этих прав. Работа содержит в себе основные понятия о файловых системах и дается общий обзор представителей, для работы с которыми выполнена программа. В статье рассматриваются основные права доступа к файлам, также представлена практическая реализация проекта.

Целью данной статьи является разработка ПО для автоматизации управления режимом доступа к ресурсам файловой системы, что позволит сократить время установки прав доступа и уменьшить количество возможных ошибок. Из поставленной цели вытекают следующие **задачи**:

- Изучение основ файловой системы Linux;
- Изучение ACL и методов работы с ним;
- Написание программного обеспечения для автоматизации работы с ACL.
- Создание графического интерфейса пользователя.
- Отладка реализованного ПО.

Ключевые слова: Файловая система, ACL, права доступа, средства разработки.

Введение

Операционная система Linux имеет ряд различных функций безопасности. Одна из них – система управления правами доступа. Linux, как последователь идеологии ядра Linux в отличие от Windows, изначально проектировался как многопользовательская система, поэтому в данной системе существуют механизмы разграничения доступа пользователей [2].

В настоящее время управление разграничением доступа к информации на файловых системах, таких как NFS и Lustre, является трудоемким процессом, который становится все сложнее и сложнее из-за роста количества пользователей, информации и задач, решаемых с помощью вычислительных комплексов. Для более гибкого управления параметрами доступа используется ACL, который имеет разные модификации на файловых системах NFS и Lustre.

Методы и принципы исследования

Общие сведения о файловых системах

Данные на диске хранятся в виде файлов. Файл - это именованная часть диска.

Для управления файлами предназначены системы управления файлами.

Возможность иметь дело с данными, хранящимися в файлах, на логическом уровне предоставляет файловая система. Именно файловая система определяет способ организации данных на каком-либо носителе данных.

Таким образом, ***файловая система*** (Рис.1) - это набор спецификаций и соответствующее им программное обеспечение, которые отвечают за создание, уничтожение, организацию, чтение, запись, модификацию и перемещение файловой информации, а также за управление доступом к файлам и за управлением ресурсами, которые используются файлами.

Рис. 1 Многоуровневая файловая система

Система управления файлами является основной подсистемой в абсолютном большинстве современных ОС.

С помощью системы управления файлами:

- связываются по данным все системные обрабатывающие программы;
- решаются проблемы централизованного распределения дискового пространства и управления данными;
- предоставляются возможности пользователю по выполнению операций над файлами (создание и т.п.), по обмену данными между файлами и различными устройствами, по защите файлов от несанкционированного доступа.

В некоторых ОС может быть несколько систем управления файлами, что обеспечивает им возможность работать с несколькими файловыми системами.

Постараемся различать файловую систему и систему управления файлами.

Термин «файловая система» определяет принципы доступа к данным, организованным в файлы.

Термин «система управления файлами» относится к конкретной реализации файловой системы, т.е. это комплекс программных модулей, обеспечивающих работу с файлами в конкретной ОС.

Файловые системы, используемые для разработки

Network File System

NFS (Рис.2) — протокол сетевого доступа к файловым системам, первоначально разработан Sun Microsystems в 1984 году. За основу взят протокол вызова удалённых процедур (ONC RPC). Позволяет подключать (монтировать) удалённые файловые системы через сеть [7].

NFS абстрагирован от типов файловых систем как сервера, так и клиента, существует множество реализаций NFS-серверов и клиентов для различных операционных систем и аппаратных архитектур. Наиболее зрелая версия NFS — v.4, поддерживающая различные средства аутентификации (в частности, Kerberos и LIPKEY с использованием протокола RPCSEC GSS) и списки контроля доступа (как POSIX, так и Windows-типов).

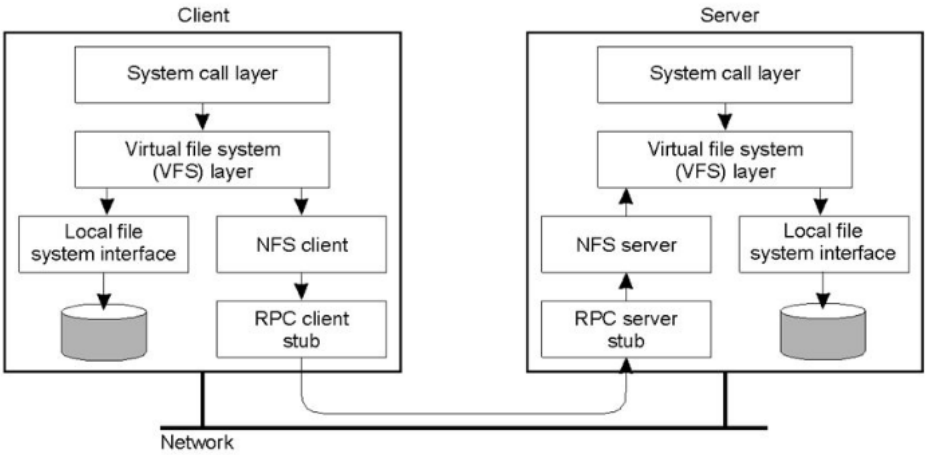

Рис.2 Классическая архитектура NFS для UNIX систем.

NFS предоставляет клиентам прозрачный доступ к файлам и файловой системе сервера. В отличие от FTP, протокол NFS осуществляет доступ только к тем частям файла, к которым обратился процесс, и основное достоинство его в том, что он делает этот доступ прозрачным. Это

означает, что любое приложение клиента, которое может работать с локальным файлом, с таким же успехом может работать и с NFS-файлом, без каких-либо модификаций самой программы.

NFS-клиенты получают доступ к файлам на NFS-сервере путём отправки RPC-запросов на сервер. Это может быть реализовано с использованием обычных пользовательских процессов — а именно, NFS-клиент может быть пользовательским процессом, который осуществляет конкретные RPC-вызовы на сервер, который так же может быть пользовательским процессом.

Требованиями при разработке NFS были:

- потенциальная поддержка различных операционных систем (не только UNIX), чтобы серверы и клиенты NFS возможно было бы реализовать в разных операционных системах;
- протокол не должен зависеть от каких-либо определённых аппаратных средств;
- должны быть реализованы простые механизмы восстановления в случае отказов сервера или клиента;
- приложения должны иметь прозрачный доступ к удаленным файлам без использования специальных путевых имён или библиотек и без перекомпиляции;
- для UNIX-клиентов должна поддерживаться семантика UNIX;
- производительность NFS должна быть сравнима с производительностью локальных дисков;
- реализация не должна быть зависимой от транспортных средств.

Lustre

Lustre— распределённая файловая система массового параллелизма, используемая обычно для крупномасштабных кластерных вычислений. Реализованный под лицензией GNU GPL, проект предоставляет высокопроизводительную файловую систему для кластеров с десятками тысяч узлов сети и петабайтными хранилищами информации [7].

Файловые системы Lustre используются в компьютерных кластерах, начиная от небольших кластеров рабочих групп и заканчивая масштабными географически распределёнными кластерами. Пятнадцать суперкомпьютеров из мирового «Топ-30» используют файловые системы Lustre.

Файловые системы Lustre могут поддерживать десятки тысяч клиентских систем, десятки петабайт (PBs) памяти для хранения данных и пропускную способность ввода-вывода в сотни гигабайт в секунду (GB/s). Благодаря высокой масштабируемости Lustre, такие области бизнеса, как провайдеры Интернет, финансовые организации, индустрия нефти и газа устанавливают файловые системы Lustre в своих центрах обработки данных.

Файловая система Lustre (Рис.3) состоит из следующих подсистем:

- **Сервер метаданных (MetaData Server (MDS))**, который хранит имена файлов и каталогов и их права.
- **Серверы для хранения самих данных (Object Storage Servers (OSSes))**. Общий объем доступного пространства Lustre вычисляется как сумма пространств на этих серверах.
- **Клиенты**. Серверы, которые имеют доступ к файловой системе Lustre.

1) MDS, OSSes и клиенты могут располагаться как и на одном сервере, так и на разных серверах. Для взаимодействия Lustre поддерживает различные протоколы, включая Infiniband, TCP/IP поверх Ethernet, Myrinet и Quadrics.

OSS поддерживает файловую систему (на данный момент – ext3, в будущем – ZFS) для хранения локальных данных, которая экспортируется в кластер для операций чтения/записи.

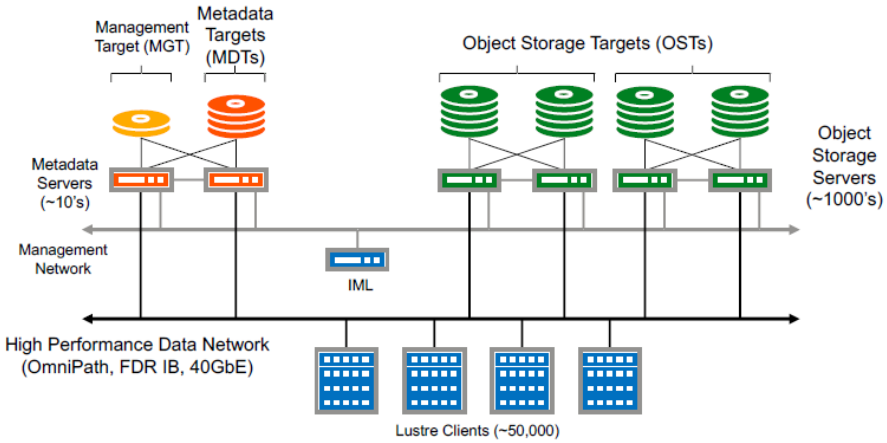

Рис.3 Представление Lustre.

Списки контроля доступа

Access Control List (Рис. 4) или ACL — список контроля доступа, который определяет, кто или что может получать доступ к конкретному объекту, и какие именно операции разрешено или запрещено этому субъекту проводить над объектом [1].

```
student@PC ~ $ setfacl -d -m u::rwx,g::r--,o::r--,u:admin:r-x Homework/
student@PC ~ $
student@PC ~ $ getfacl Homework/
# file: Homework/
# owner: student
# group: student
user::rwx
group::r--
other::r--
default:user::rwx
default:user:admin:r-x
default:group::r--
default:mask::r-x
default:other::r--
```

Рис.4 ACL по умолчанию.

По умолчанию работа с правами доступа в Linux довольно примитивна. Можно использовать только три категории для управления правами доступа: отдельно права для владельца файла, группы владельца и всех остальных пользователей. Этого вполне достаточно при работе с обычным домашним компьютером, где нет большого количества учетных записей пользователей и они не обмениваются между собой файлами. Но если нужно настроить систему большой организации с огромным количеством пользователей и потребностью обмена файлами по Samba и NFS понадобиться более сложная конфигурация.

Использование списков контроля доступа позволяет более гибко контролировать доступ к файлам, которыми вы хотите поделиться с другими пользователями.

Существуют два типа ACL:

- ACL для доступа;
- ACL по умолчанию.

ACL для доступа — это список управления доступом для заданного файла или каталога. Проще говоря - это сами права на объект, которые будут контролировать доступ к этому объекту.

ACL по умолчанию - может быть связан только с каталогом, и, если файл в этом каталоге не имеет ACL для доступа, он использует правила, определённые в ACL по умолчанию, связанном с каталогом.

Управления ACL списками осуществляется двумя командами:

1. setfacl - используется для назначения, модификации и удаления ACL прав.

2. getfacl - используется для просмотра установленных ACL.

Основные результаты

Далее представлены основные фрагменты реализация проекта.

Для полного понимания работы предоставлена ссылка на целый листинг программы [3]

В основе разрабатываемого ПО лежит класс, отвечающий за все события в коде:

```
class User:

def __init__(self, name):
self.name = name
self.privileges = {}

def add_privilege(self, path, permission):
self.privileges[path] = permission

def get_privileges(self):
return self.privileges

def get_name(self):
return self.name

def __str__(self):
return self.name
```

Рассмотрим основные методы:

- __init__(self, name) – Конструктор класса. При вызове класса в круглых скобках передаются значения, которые будут присвоены параметрам метода __init__(). Первый его параметр – self – ссылка на сам только что созданный объект.
- add_privilege(self, path, permission) – выполняет функцию добавления прав.
- get_privileges(self) – выполняет функцию проверки прав доступа пользователя.
- get_name(self) – узнает имена пользователей.

__str__(self) - вызывается функциями str, print и format. Возвращает строковое представление объекта.

Реализация программы выполнена в двух вариациях: 1) консольная, 2) графический интерфейс пользователя. Далее представлен второй вариант, так как именно с ним будет работать конечный пользователь.

Графический интерфейс (Рис.5) реализуется с помощью PyQT5 и имеет вид:

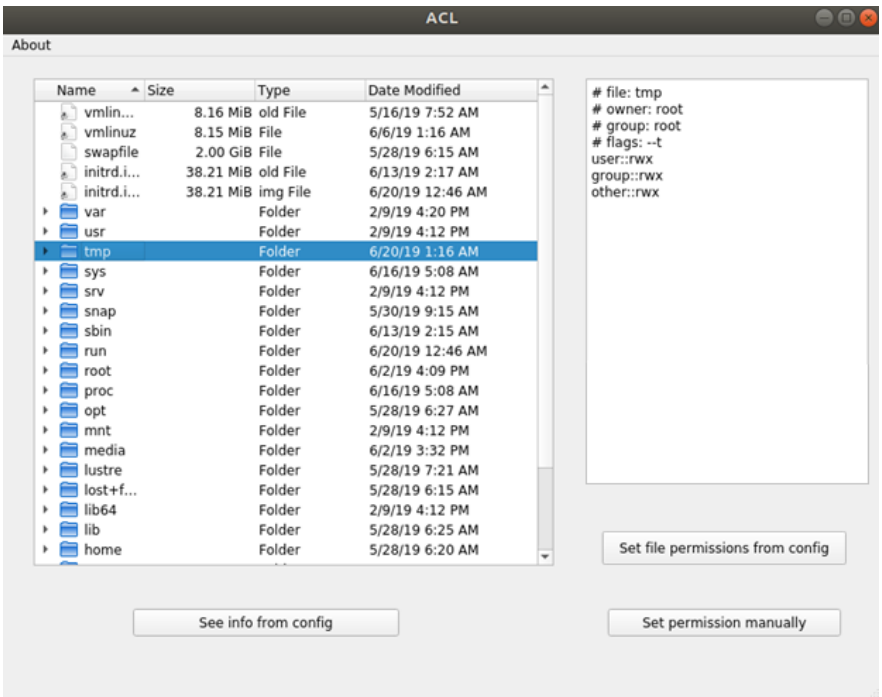

Рис. 5 Графический интерфейс пользователя

Описание функционала программы:

Кнопка «Set file permissions from config» - выполняет основную программу и применяет указанные в конфигурационном файле значения.

Программа собирает всю информацию из файла conf (Рис.6):

```
1  /lustre/test:group3,group1:rwx
2  /lustre/test1:group2:rw
3  /lustre/test2:group3:r
4  /lustre/test3:group1:rwx
5  /lustre/test4:group2:rwx
6  /lustre/test5:group3:rw
```

Рис.6 Содержимое файла conf

Кнопка «Set permission manually» - вызывает часть ниже представленного кода (Рис.7) на изменение вручную и соответственно дает возможность пользователю изменить права доступа, которые не указаны в конфигурационном файле.

```
start = 'user::rx\nuser:test:rx\ngroup::rwx\n'
# how middle line should look like
# '\ngroup:{gr}:{pr},group:{gr}:{pr}, .. \nmask
end = 'mask::rx\nother::rx\n'

for line in conf:
    middle = ''
    # groups for this line of config


    # left and right colons
    lc = line.find(':')
    rc = line.rfind(':')

    # parse line and get groups
    groups = line[lc+1:rc].split(',')

    # get permission
    perm = line[rc+1:]
    perm = perm.rstrip('\n')

    # form middle line using multiple groups and permission
    for group in groups:
        middle = middle + 'group:{gr}:{p},'.format(gr=group, p=perm)

    middle = middle.rstrip(',')
    middle = middle + '\n'

    path = line[:lc]
    s = start + middle + end
    a = posix1e.ACL(text=s)
    a.applyto(path)
```

Рис.7 Код, вызываемый на изменения вручную

Кнопка «See info from config» - выводит на экран результат выполнения представленной в консольной реализации функции «print_users_info».

Кнопка «About» - выводит пользователю краткое описание, представленное выше.

Также при нажатии правой кнопки мыши по любому файлу вызывается контекстное меню (Рис.8) с возможностью увидеть права доступа к этому файлу и открыть его.

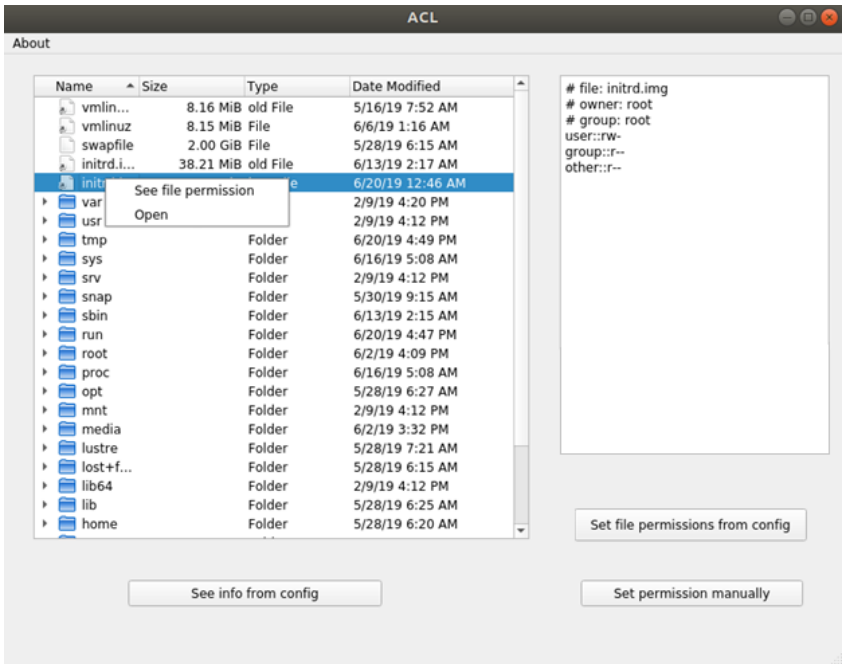

Рис.8 Контекстное меню

Заключение

В ходе выполнения работы были изучены основы файловых систем, а также прав доступа к файлам в Linux.

В данной работе были освоены способы работы с ACL, реализовано программное обеспечение для автоматизации управления режимом доступа к ресурсам файловой системы.

Использование библиотеки Qt позволило сделать приложение с дружественным интерфейсом пользователя. Интерфейс позволяет пользователю получать больше информации в более удобном виде, а также эффективнее взаимодействовать с программой.

Цель, поставленная в рамках работы, достигнута и задачи, вытекающие из нее, выполнены.

Список литературы

1. Access control list linux [Электронный ресурс]. – Режим доступа: https://help.ubuntu.ru/wiki/access_control_list

2. Linux для пользователя [Электронный ресурс]. – Режим доступа: http://www.linuxcenter.ru/

3. Листинг программы [Электронный ресурс]. – Режим доступа: https://docs.google.com/document/d/1sT_b7W9FNwAQlFfp_3ribFeHKGBLD3fCdjA5aMfc_6o/edit#

4. Права доступа к файлам в Linux [Электронный ресурс]. – Режим доступа: https://losst.ru/prava-dostupa-k-fajlam-v-linux

5. Системы управления файлами [Электронный ресурс]. – Режим доступа: http://mf.grsu.by/UchProc/livak/b_lecture/lec33_SYF.htm

6. Скотт Граннеман. Linux. Карманный справочник. – Москва: Вильямс, 2018 – 464 с.

7. Типы файловых систем для Linux [Электронный ресурс]. – Режим доступа: https://losst.ru/tipy-fajlovyh-sistem-dlya-linux

8. Уильям Шоттс. Командная строка Linux. Полное руководство. – П: Питер, 2017 – 480 с.

9. Эндрю Таненбаум, Х. Бос. Современные операционные системы. –П: Питер, 2017 – 1120 с.