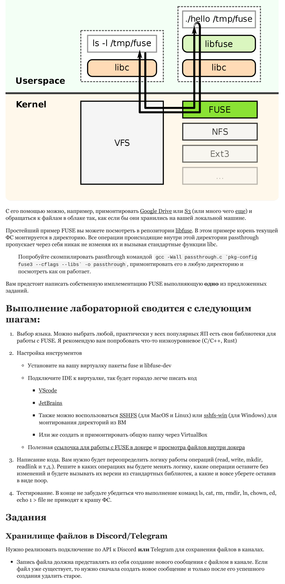
Лабораторная #6. Работа с модулем FUSE

FUSE (Filesystem in userspace)

FUSE представляет собой набор инструментов для монтирования ФС в пользовательском пространстве.



С его помощью можно, например, примонтировать Google Drive или S3 (или много чего еще) и обращаться к файлам в облаке так, как если бы они хранились на вашей локальной машине.

Простейший пример FUSE на можете посмотреть в репозитории libfuse. В этом примере корень текущей ФС монтируется в директорию. Все операции происходящие внутри этой директории passthrough проксируют через себя никак не изменяя их и вызывая стандартные функции libc.

> Попробуйте скомпилировать passthrough командой `gcc -Wall passthrough.c `pkg-config fuse3 --cflags --libs` -o passthrough`, примонтировать его в любую директорию и посмотреть как он работает.

Вам предстоит написать собственную имплементацию FUSE выполняющую одно из предложенных заданий.

Выполнение лабораторной сводится к следующим шагам:

1. Выбор языка. Можно выбрать любой, практически у всех популярных ЯП есть свои библиотеки для работы с FUSE. Я рекомендую как попробовать что-то низкоуровневое (C/C++, Rust)
2. Настройка инструментов
   - Установите на вашу виртуалку пакеты fuse и libfuse-dev
   - Подключите IDE к виртуалке, так будет гораздо легче писать код
     - VScode
     - JetBrains
     - Также можно воспользоваться SSHFS (для MacOS и Linux) или sshfs-win (для Windows) для монтирования директорий из ВМ
     - Или же создать в примонтированную общую папку через VirtualBox
   - Полезная ссылочка для работы с FUSE в докере и просмотра файлов внутри докера
3. Написание кода. Вам нужно будет переопределить логику работы операций (read, write, mkdir, readlink и т.д.). Решите в каких операциях вы будете менять логику, какие операции оставите без изменений и будете вызывать их версии из стандартных библиотек, а какие и вовсе уберете оставив в виде ноор.
4. Тестирование. В конце не забудьте убедиться что выполнение команд ls, cat, rm, rmdir, ln, chown, cd, echo и т. д. не приводит к краху ФС.

Задания

Хранилище файлов в Discord/Telegram

Нужно реализовать подключение по API к Discord **или** Telegram для сохранения файлов в каналах.

- Запись файла должна представлять из себя создание нового сообщения с файлом в канале. Если файл уже существует, то нужно сначала создать новое сообщение и только после его успешного создания удалить старое.
- Директории
  - В Telegram при переименовании файла в директории к сообщению нужно добавлять тег с названием директории. При листинге директории нужно отдавать список файлов с тегом. Удаление директории должно приводить к удалению всех файлов с соответствующим тегом.
  - Для Discord запись файлов в корневую директорию должна сохранять файлы в основной текстовый канал. При создании директории должен создаваться новый канал и файлы записанные в новую директорию должны загружаться в этот канал. При листинге директории нужно отдавать список файлов из канала. При удалении директории канал должен удаляться.
  - Создание вложенных директорий реализовывать не нужно
- Запрос ctime и mtime должен возвращать дату создания сообщения с файлом
- Остальные операции (смена владельца файла, создание ссылок) можно не реализовывать

Для хранения мета информации можно как использовать локальный файл, так и специальное сообщение в канале

Конвертация файлов на лету

В вашу программу будет передаваться путь до директории с картинками в png формате, отсуда она будет получать изначальные файлы.

- При листинге директории мапа программа должна отдавать список оригинальных файлов и (если применимо) их сконвертированные копии. Пример:

```
.
├── mounted_dir
│   ├── not_pic.txt
│   ├── screenshot.jpg
│   ├── screenshot.png
│   ├── tux.jpg
│   └── tux.png
└── original_dir
    ├── not_pic.txt
    ├── screenshot.png
    └── tux.png
```

- При попытке открыть файл jpg оригинальный png должен конвертироваться и возвращаться в jpg формате. Для того чтобы сбрасывать что-фай действительно сконвертировался можно воспользоваться утилитой file
- Все остальные операции должны перенаправляться на оригинальную директорию. Например mkdir mounted_dir/new_dir создаст директорию в original_dir.
- При желании с преподавателем можно обсудить другие форматы для конвертации, например flac->mp3 или md->pdf

Группировка mp3 файлов

У mp3 файлов есть теги содержащие разную мета информация об исполнителе, названии и многом еще, год написания песни, жанр и т.д. Ваша программа будет получать путь до директории с mp3 файлами и должна будет прийти рекурсивно по всем поддиректориям составить у директорий в которых будут сгруппированы файлы согласно тегам: Artist, Year и Genre.

Пример:

```
.
├── grouped_mp3
│   ├── Artist
│   │   ├── Gary Jules
│   │   │   └── mad_world-Gary_Jules.mp3
│   │   ├── Lofi Girl
│   │   │   └── lofi_beats_to_relax_study_to.mp3
│   │   ├── Mick Gordon
│   │   │   └── mick gordon RIP&TEAR.mp3
│   │   └── no_artist
│   │       └── Без названия - Неизвестен.mp3
│   ├── Genre
│   │   ├── Hip-Hop
│   │   │   └── lofi_beats_to_relax_study_to.mp3
│   │   ├── Metal
│   │   │   └── mick gordon RIP&TEAR.mp3
│   │   ├── New Wave
│   │   │   └── mad_world-Gary_Jules.mp3
│   │   ├── no_genre
│   │   │   └── Без названия - Неизвестен.mp3
│   │   ├── Soundtrack
│   │   │   └── mick gordon RIP&TEAR.mp3
│   │   └── Synthpop
│   │       └── mad_world-Gary_Jules.mp3
│   └── Year
│       ├── 1982
│       │   └── mad_world-Gary_Jules.mp3
│       ├── 2016
│       │   └── mick gordon RIP&TEAR.mp3
│       └── no_year
│           ├── lofi_beats_to_relax_study_to.mp3
│           └── Без названия - Неизвестен.mp3
└── original_dir
    ├── cool_vibes
    │   └── lofi_beats_to_relax_study_to.mp3
    ├── mick gordon RIP&TEAR.mp3
    └── музыка из ам
        ├── mad_world-Gary_Jules.mp3
        └── Без названия - Неизвестен.mp3
```

- ФС должна быть read only
- Обратите внимание что у некоторых файлов может быть несколько жанров
- При попытке прочитать файл из примонтированной директории должен отдаваться оригинальный файл

Авторазархивирование

В вашу программу будет передаваться путь до директории с разными файлами, в том числе архивами. Для каждого архива в примонтированной ФС или необходимо показывать директорию с содержимым архива и возможностью прочитать содержимое файлов.

Пример поведения ФС:

```
zip_task$ tree

├── mounted_dir
└── original_dir
    ├── archive.tar.gz
    ├── labs.zip
    └── not_archive.txt
zip_task$ ./zip_task original_dir mounted_dir
zip_task$ tree

├── mounted_dir
│   ├── archive
│   │   ├── another_file
│   │   └── some_file
│   ├── archive.tar.gz
│   ├── labs
│   │   ├── lab_1.pdf
│   │   ├── lab_2.pdf
│   │   └── readme.txt
│   ├── labs.zip
│   └── not_archive.txt
└── original_dir
    ├── archive.tar.gz
    ├── labs.zip
    └── not_archive.txt
zip_task$ cat mounted_dir/labs/readme.txt
Лабы по ОС
```

- Архивные директории (archive, labs) должны быть read only
- Все остальные операции должны перенаправляться на оригинальную директорию. Например mkdir mounted_dir/new_dir создаст директорию в original_dir
- Ваша имплементация должна поддерживать минимум 2 архивных формата на ваш выбор

ФС с кастомными правилами

Ваша задача реализовать ФС которая будет в зависимости от конфига и типа файла исполнять разные операции.

Пример формата конфига:

```
filetypes:
  - mp3:
      read: pandoc {{ filename }} -o {{ filename_without_extension }}.pdf
  - png:
      write: exit 0 # don't change or create png files
      read: cat /root/default.png # return default picture for all png
  - other:
      read:
        - pass # do what usually do when read file
        - echo ${date} || filename }} >> /var/log/fs_log
DIRECTORY_LISTING:
  - echo "Total files ${find | -type f | wc -l}"
  - echo "Total directories ${find | -type d | wc -l}"
  - echo "Total hidden ${ls -a | -type d | grep -E '^\.*$' | wc -l}"
  - pass
```

Формат вашего конфига не обязательно должен быть в yaml формате и не обязательно именно с такими названиями полей. Но там должны быть:

- Маски с расширениями файлов
- У перечисленных расширений должен быть поле read или write где описываются операции которые должны быть выполнены при чтении или записи файла с определенным расширением
- Поля read/write могут быть строкой или массивом если планируется делать несколько операций с файлом
- Переменные `filename` и `filename_without_extension` которые будут подставляться при выполнении команд
- Специальное расширение other обозначающее все остальные расширения файлов кроме перечисленных
- Специальная строка pass обозначающая операцию по умолчанию. В примере все файлы с расширением не mp3 или png будут открываться как обычно, а затем логироваться информация об этом
- Поле DIRECTORY_LISTING где будут команды, исполняемые при листинге директории (`readdir`)

Пример:

```
custom_rules$ ls -l
Total files 4
Total directories 0
Total hidden 0
-rw-r--r--    4 vlad 24 апр 00:01 dont_readme.txt
-rw-r--r-- 3275 vlad 12 фев  2021 pic.png
-rw-r--r--  15k vlad 24 апр 00:02 "Лабораторная #6. Работа с модулем FUSE.md
custom_rules$ echo 123 > pic.png
custom_rules$ ls -l pic.png # Размер не изменился, т.к. запись не прошла
-rw-r--r-- 3275 vlad 12 фев  2021 pic.png
custom_rules$ cat Лабораторная\ \#6.\ Работа\ с\ модулем\ FUSE.md
custom_rules$ ls -l
Total files 4
Total directories 0
Total hidden 0
-rw-r--r--    4 vlad 24 апр 00:01 dont_readme.txt
-rw-r--r-- 3275 vlad 12 фев  2021 pic.png
-rw-r--r--  15k vlad 24 апр 00:02 "Лабораторная #6. Работа с модулем FUSE.md'
-rw-r--r-- 1019 vlad 24 апр 00:03 "Лабораторная #6. Работа с модулем FUSE.pdf'
custom_rules$ cat dont_readme.txt
123
custom_rules$ cat /var/log/fs_log
Ср 24 апр 2024 00:00:57 MSK dont_readme.txt
```

Свое задание

Если у вас есть другая идея с использованием FUSE и вы хотите реализовать ее, то обсудите это с преподавателем чтобы расписать ТЗ в вашему заданию.