Задание

— Добавить поддержку директорий.
— Добавить поддержку символических ссылок.
— Добавить права доступа к файлам по аналогии с chmod (RWX), считая, что у нас только один пользователь.
— Реализовать утилиты для работы с файловой системой (mkdir, rm <с поддержкой удаления директории>, ln, chmod).
— Реализовать устройства консоли: /dev/stdout и /dev/stdin в которые/из которых позволит читать/писать в консоли.
— Написать тесты для проверки корректности добавленной реализации.

Техническая реализация

- **Директории**

Была реализована консольная утилита mkdir [name], позволяющая создавать директории. Директории создаются вызовом mkdir (определена в dir.c), которая создает файл со специальным флагом O_MKDIR. В файлах созданных с таким флагом f_type устанавливается в значение FTYPE_DIR. Основная логика работы с директориями была реализована в jos до нас.
Нами была добавлена возможность изменения текущей директории процесса за счет специального вызова sys_env_set_workpath и соответствующего поля в struct Env, а также консольная утилита cd [path] выполняющая это действие и утилита pwd, которая выводит текущее значение workpath.

- **Символические ссылки**

Была реализована консольная утилита ln [file] [link], позволяющая создавать символические ссылки на файлы. Символические ссылки создаются вызовом symlink (определена в file.c), которая создает файл со специальным флагом O_MKLINK и записанным в него путем к файлу на который ссылается ссылка. В файлах созданных с таким флагом f_type устанавливается в значение FTYPE_LINK. При открытии файлов с данным типом из файла читается путь к файлу и подставляется в file_open вместо пути ссылки.
Для того чтобы записать путь в файл используется флаг O_SYSTEM, который может использоваться только системой при создании ссылки. При любом чтении/записи/выполнении без этого флага происходит переадресация на файл, на который ссылается данная ссылка.

- **Права доступа**

В заголовке каждого файла присутствует поле f_perm с правами доступа для одного пользователя от 0 до 7, как у chmod в UNIX системах. Была реализована утилита chmod [permissions][file], позволяющая менять права доступа у файла. Права файла изменяются вызовом chmod (определена в file.c), которая открывает файл с флагом O_CHMOD и значением желаемых прав доступа, передаваемых через req_omode со смещением 0x4, далее за счет вызова

file_set_perm в struct File меняются права доступа. Ограничения на открытие файлов в режиме чтения/записи/выполнения реализованы в функции serve_open (определена в serv.c).
Ограничение на выполнение реализовано за счет введения нового флага O_SPAWN который применяется только в функции spawn.
Также были добавлены макросы PERM_READ, PERM_WRITE, PERM_EXEC, упрощающие работы с правами файлов.

- **Поддержка удаления файлов и директорий**

Была реализована консольная утилита rm [file], удаляющая файл или директорию. Удаление происходит за счет вызова remove (определена в file.c), которая посылает ipc сигнал FSREQ_REMOVE с путем к выбранному файлу, при получении такого сигнала сервер файловой системы вызывает serve_remove (определена в serv.c), которая вызывает file_remove (определена в fs.c). Данная функция изменяет размер файла на 0 и удаляет данный файл из той директории в которой он находится. Если же удаляемый файл является директорией, то помимо этого из этой директории удаляются все файлы.

- **Устройство консоли /dev/stdout, /dev/stderr и /dev/stdin**

Для реализации возможностей перенаправления потоков ввода, вывода и ошибок в файлы и из файлов в эти потоки были реализованы специальные файлы. Данные файлы создаются процессом init при запуске системы. При чтении/записи в эти файлы функция open возвращает 0, 1 или 2 дескрипторы, что позволяет открыть и использовать соответствующий поток.
Поддержка потока ошибок была реализована по аналогии с поддержкой потока вывода в файле init.c.

- **Прочие утилиты для работы с файлами**

Реализована утилита touch для создания пустого файла. Что позволило облегчить тестирование.

Тестирование

Целью тестирование является выявление существующих багов и багов в ходе дальнейшей модификации исходного кода. В ходе тестирования были выявлены такие недостатки, как невозможность открыть на запись файл через symlink. Также были устранены такие возможности, как открытие директории как файл на чтение, запись и выполнение. Также был найден баг связанный с повторным созданием директорий.

Суть тестирования заключается в проверке корректности работы каждой реализованной функции отдельно.

- Проверка корректности директорий: заключается в создании директории, проверка корректности при работе с ней (отсутствие возможности открыть ее, как файл на чтение, запись, исполнение). Удаление директории, проверка что содержимое директории действительно удалено.
- Проверка корректности работы символических ссылок: заключается в создании файла, записью в него некоторого значения, после чего создается symlink на этот файл. Через ярлык осуществляется попытка записи и чтения. После чего проверяется корректность полученных результатов. В конце проверяется корректность поведения ярлыка при удалении файла на который он ссылается. Также проверяется возможность создания ярлыка на директорию. Тесты связанные с этой возможностью схожи с тестами, проведенными с директорией.
- Проверка прав доступа: заключается в создании файла, после его создания файлу меняются права и через механизм fstat проверяется корректность присвоенных прав. После чего проверяется действительно ли у этого файла больше нету таких прав (попыткой открыть его на чтение, запись или исполнение).
- Проверка устройства консоли: заключается в записи (чтении) данных в нее напрямую, после чего данные читаются (записываются) в соответствующий файловый дескриптор. В итоге записанные (прочитанные) данные сравниваются с прочитанными (записанными).

Запустить написанные тесты можно командой testfile в консоли, либо с помощью make run-testfile.

https://docs.google.com/document/d/1wLv6xGMCtt7tdldCPCELgXCGXeeZ-uqt5ycY-n6_cM0/edit?usp=sharing