1. Поддержка директорий
    - В структуру Env (информация о процессе, inc/env.h) добавлено поле char workpath[MAXPATHLEN]; MAXPATHLEN = 1024 (константа была задана в JOS) - максимальная длина пути. Поле хранит путь текущей директории процесса.
    - Тип файла-директории FTYPE_DIR = 1 уже был реализован в JOS (inc/fs.h).
    - Утилиты:
        - pwd. Выводит текущую рабочую директорию процесса. Для её работы в lib/dir.c реализована функция getcwd, которая запрашивает workpath поле текущего процесса.
        - cd. Меняет текущую рабочую директорию процесса. Для её работы в lib/dir.c реализована функция chdir. В этой функции по входному пути файл проверяется на тип директории, путь форматируется (убираются “.”, “..”, дублирующиеся “/”), в конце выполняется системный вызов SYS_env_set_workpath, который меняет у процесса поле workpath.
        - mkdir. Создаёт новую директорию с заданным именем. Для её работы в lib/dir.c реализована функция mkdir. Вызывает open для создания файла с mode = O_MKDIR | O_EXCL (тип - директория и защита от создания директории с занятым именем). Далее эта функция посылает IPC-сигнал на открытие файла, он обрабатывается serv/serve_open, далее вызывается fs.c/dir_create, по аналогии с процессом создания файла.
        - rm. Удаляет файл или директорию. Для её работы в lib/file.c реализована функция remove. В этой функции заданный путь форматируется (убираются “.”, “..”, дублирующиеся “/”), посылается IPC-сигнал на удаление файла, он обрабатывается в serv/serve_remove, далее вызывается fs.c/remove_analyze. Эта функция, если ей подали директорию, обходит её рекурсивно и посылает в fs.c/file_remove файлы и директории на удаление.
2. Символические ссылки
    - Тип символической ссылки FTYPE_LINK = 2 добавлен в inc/fs.h
    - Добавлен флаг O_SYSTEM = 0x4000, который может использоваться только системой при создании ссылки.
    - Добавлен флаг O_MKLINK = 0x2000 для
    - 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. При любом чтении/записи/выполнении без этого флага происходит переадресация на файл, на который ссылается данная ссылка.

3. Права доступа
    - Добавлен флаг O_CHMOD = 0x1000 для работы с chmod.
    - В заголовке каждого файла (struct File) присутствует поле f_perm с правами доступа для одного пользователя от 0 (000) до 7 (111), по аналогии с UNIX системами.
    - 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 (т.е. занимают второй байт), далее за счет вызова далее за счет вызова из serv.c/serve_open функции fs.c/file_set_perm в struct File меняются права доступа (поле f_perm). Ограничения на открытие файлов в режиме чтения/записи/выполнения реализованы в функции serve_open (определена в serv.c). Флаг O_SPAWN, ставящийся в функции spawn.c/spawn, позволяет сделать ограничение на выполнение.
4. Устройство консоли
    - 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.
    - Поддержка pipe в консоли была реализована в JOS до нас.
5. Дополнительные утилиты для работы с файлами
    - Была реализована утилита touch для создания пустого файла. Эта утилита используется в тестировании. Принцип работы - создаёт в текущей директории файл с указанным именем (открывает файл с модом O_CREAT | O_EXCL). Нельзя создать файл с занятым именем.
6. Тесты. Каждую функциональность проверяли тестами в тестовом файле. Запуск тестирования из Shell: testfile.
    - Поддержка директорий. Создаём директорию и проверяем работу с ней (запрещено открыть ее как файл на чтение/запись/исполнение). Удаление директории, проверка, удалилось ли её содержимое.
    - Символические ссылки. Создаём файл, записываем в него данные, создаём symlink на этот файл. Через symlink пытаемся записать и прочитать, проверяем результат, корректность поведения symlink’a при удалении файла, на который он ссылается, возможность создания symlink’a на директорию.
    - Права доступа. Создаём файл, изменяем права доступа, с помощью fstat проверяем корректность этих прав, проверяем отсутствие/присутствие прав после их изменения попыткой открыть его на чтение/запись/исполнение.
    - Устройство консоли. Записываем (читаем) данные в (из) консоль, читаем их (записываем) в соответствующий файловый дескриптор, записанные (прочитанные) данные в консоль сравниваем с прочитанными (записанными) из дескриптора.