Файлова система върху единичен файл

Условие

Да се реализира програма, която симулира файлова система върху един двоичен файл (контейнер). Структурата на контейнера е по преценка на студента. Програмата трябва да получава параметрите си от командния ред. Директориите могат да се влагат с произволна дълбочина. Не е задължително контейнерът да може да се разширява неограничено много, достатъчно е при създаването му да се окаже максимален размер.

Контейнерът не трябва да се зарежда в оперативната памет при промяна, а директно да се модифицира файла, в който е реализиран контейнера[1].

Трябва да се поддържат следните функционалности:

1. Основни операции

- Създаване на директория (**mkdir**);
- Изтриване на празна директория (**rmdir**);
- Извеждане на съдържанието на директория (**ls**);
- Промяна на текущата директория (**cd**);
- Копиране на файл (**cp**);
- Изтриване на файл (**rm**);
- Извеждане на съдържанието на файл на екрана (**cat**);
- Записване на съдържание към файл (**write**). Ако не е подадена допълнителната опция +**append**, се създава празен файл (ако такъв съществува се презаписва). Ако такава опция е подадена и се записва върху съществуващ файл, тогава съдържанието се добавя в края му. Съдържанието се подава като текст ограден в кавички - параметър на командата.
- Копиране на файл от външна файлова система във вашата (**import**). Тук подавате път към реален файл (**source**) и към файл от вашата файлова система (**destination**). Трябва да се поддържа параметър +**append**, с действие подобно на това при командата write.
- Копиране на файл от вашата файлова система на външна (**export**).

[1] Реализацията на масив и свързан списък в поток е разгледана в У3 и У4.

2. Deduplication.

Файловата система трябва да разбива всеки от съхранените в нея файлове на един или повече блокове с размер N. Този размер се подава от потребителя при създаването на контейнера и не може да се променя след това. Реализирайте схема за премахване на дублиране (deduplication), която гарантира, че ако даден блок се използва в повече от един файл, той се пази само веднъж във файловата система.

3. Resiliency.

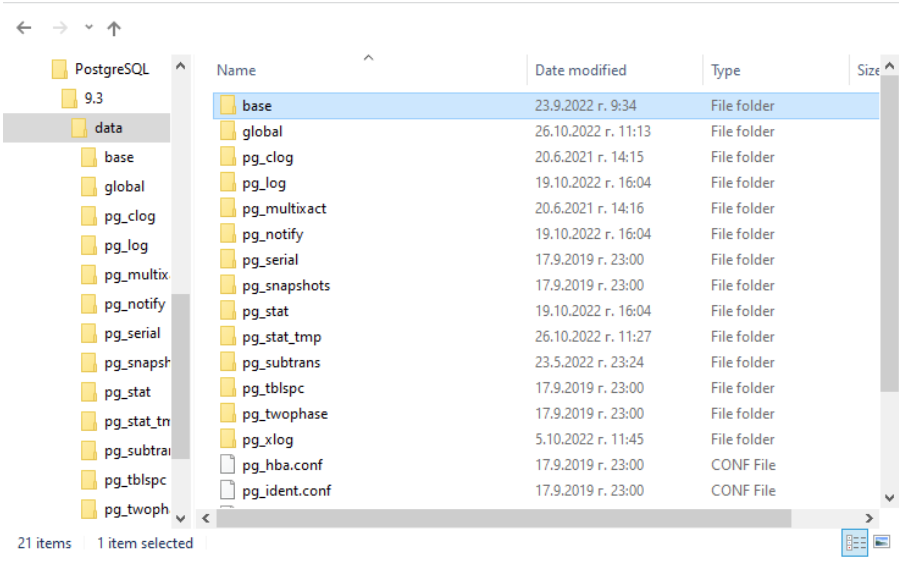
Реализирайте схема за устойчивост (resiliency). Файловата система трябва да съхранява в себе си подходяща информация, с която да може да провери дали даден блок не е бил повреден (например при възникване на лош сектор на диска) след неговото записване. При всеки запис/четене на файл, resiliency информацията трябва да се обновява/проверява. Ако бъде открита повреда в даден блок, текущата операция трябва да приключи с грешка.

Трябва да обърнете внимание, какво би се случило, ако изпълнението на кода ви спре по средата на дадена операция. Пример: ако копирате голям файл и програмата бъде спряна по средата на тази операция, как можете да гарантирате, че контейнерът ще може да продължи да се използва? Разбираемо е, че файлът, който е бил в процес на създаване не е валиден и не може да се очаква с него да може да се работи. Но останалото съдържание не трябва да се загуби и не е допустимо контейнерът да стане неизползваем и да се загуби цялата информация в него.

4*. Графичен интерфейс.

Реализиране на функции за програмен достъп (API) в подходяща библиотека, позволяващи изпълнение на командите описани по-горе. Реализиране на графичен интерфейс, работещ с програмния интерфейс, позволяващ разглеждане и промяна на контейнера. Могат да се използват стандартни елементи на графичния интерфейс, като ListView, TreeView, Menu, StatusStrip и др.

Пример:



Фиг. 1. Microsoft Explorer, позоляващ визуализация и работа с файлова система, чрез употреба на ListView и TreeView.

Реализация и точки

Всички функции за обработка на текст трябва да се реализират от студента (не е разрешено използването на функциите string.Split, string.IndexOf, Regex, функциите на LINQ и т.н.). Всички помощни структури и типове трябва да се реализират от студента, в това число стекове, свързани списъци, хеш таблици, дървета и т.н.

Студентът трябва да реализира програмата в следната задължителна последователност:

1. Основни операции.

   Макс. брой точки за реализация: 30;

2. Deduplication.

   Макс. брой точки за реализация: 15;

3. Resiliency.

   Не е разрешено използването на готови хеширащи функции. Ако се използва стандартна такава тя трябва да се реализира от студента.

   Макс. брой точки за реализация: 15;

4. * - Тази точка е незадължителна. При реализиране на всички точки, вкл. незадължителната, студентът ще бъде освободен от изпит с отлична оценка.