CPSC 3220 Assignment 3

Designing a better file system “great file system”, or “grtfs” with read/write permissions.

You can work with one other teammate or by yourself. Teammates share the grade.

Problem Statement

In this assignment you will redesign the existing trivial file system. This file system served us well for a while, but with technology making rapid progress it is way too small and inefficient now. We need to have more storage space in our file system and include more useful features. You will also need to implement the three functions that are not implemented: read, write and delete.

You will find details about the old file system in the header file tfs.h. The file tfs_1.c is the first part of the implementation of helper functions and public functions. The file tfs_2.c is a skeleton file in which the implementation of the three functions (delete, read and write) would go in the future. The tfs_2.c file has header comments that specify what those functions do. The file tfs_driver1.c is an example of a test driver for the tfs_1.c and tfs_2.c code.

Your job is to increase the size of the file system to consist of 4096 blocks. The size of the blocks will be the same – 128 B. We also need a larger FAT that will contain enough entries to address all the blocks in the file system. We will have a directory with 32 Byte entries instead of 16B. You will add permission bits to write and read. Some other of the 32 B in the struct will remain unused and will be used in the future. You will need to figure out how many directory structs and FAT entries you file system will have. This is a simple math based on your understanding of how different components of the FS fit together.

You will need to rename your file system to grtfs (“great fs”), since we hope to great performance in the modified file system. This means you will need to modify all the function names, their implementations and file names to reflect the changes. You will also need to provide your own *makefil*e that allows to both make and make clean.

Implementing file-level permissions

After you make sure your write, read and delete functions work correctly, you will need to modify your program. Devise a clever way to implement file-level (not block level) read and write permissions. We are only interested in read and write permissions, we will *not* have execute permissions. If a file has a read permission set, then file can be read from. If it has a write permission set, then it can be written to. A file can have both read and write permissions set, or neither of them. When a file is created, both permissions are false by default. When the file is written to the first time, then permissions will need to be set to allow writing. When the file is read for the first time, you would need to set the read permission to allow reading. You can delete any file, no matter what its permissions are. You do not need to check permissions while deleting. We will NOT be implementing different groups of users (as in Linux) with different permissions for each in this assignment.

Write functions named *file_is_readable()* and *file_is_writable()* that take a file name and return a boolean (true if readable/writable, false otherwise). Another set of functions will be void functions *make_readable()* and *make_writable()* that take a filename as a parameter and set the corresponding permission bits. You can add whatever other helper functions you need.

Modify one of the drivers to demonstrate that your added functions work and name it tfs_driver3.c. Credit on this item will depend on how persuasive you are that your added functions work.

Submission

You will need to submit to Canvas your entire program as a compressed archive (tar.gz) named asg3.tar.gz that does not contain any subdirectories. You will have to add code to your driver to demonstrate that the larger file system is indeed functional. All parts of your submission must unzip/compile/run on the School of Computing machines.

NB: This seems like a simple assignment, since you are mostly modifying the existing code, but you will need to spend some time understanding the structure and implementation of the existing code. This is a big part of the assignment. After you understand what is going on and how different files interoperate, then you will be able to make changes to the code. Please make a plan and start right away. With the deadline close to the end of the semester, there will not be any time for assignment extension.

Grading Rubric will be posted on Canvas.

Note: Forgetting to submit your work due to any reason, corrupted submission or a submission that does not unzip/compile/run on SoC machines will receive 0 points. There will be no exceptions to the rule. Please recompile after making the last moment changes to make sure it still works and make sure you made a submission.

Guidelines

The entire code (except the parts that were provided) should be written by yourself and teammate (if working with one other person).

You may discuss the project requirements and the concepts with me or with anyone in the class.

However, you should not send code to anyone or receive code from anyone, whether by email, printed listings, photos, visual display on a computer/laptop/cell-phone/etc. screen, or *any other* method of communication.

Do not post the assignment, or a request for help, or your code on any web sites.

The key idea is that you shouldn't short-circuit the learning process for others once you know the answer. (And you shouldn't burden anyone else with inappropriate requests for code or "answers" and thus short-circuit your own learning process.)