CPSC 3220 Assignment 3

Designing a better file system named “btrfs”.

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. There are three functions that were not implemented, and you do not need to implement them at this point.

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 512 blocks. The size of the blocks will be the same – 128B. 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 16 entries. Each directory entry will increase to 32 B.

You will add the following entries to the directory_entry struct: file-level (not block level) permissions to read, write and execute the file, and a bit to indicate whether the file is encrypted or not. As usual, 1s will indicate that read, write or execute are permitted, and 1 will indicate that the file in encrypted. All of these will be unsigned character types. You will also change first_block type to unsigned short. You do not need to provide any functions to check or set any of the permission bits, or to check/encrypt/decrypt the files. You will also increase the length of the filename to 16 bytes. Some of the 32 B in the struct will remain unused and will be used in the future.

You will need to rename your file system to btrfs (“better fs”), since we hope to get better 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.

You will need to submit to canvas your entire program as a compressed archive (tar.gz) that does not contain any subdirectories. 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.

Extra Credit to be posted.

Grading Rubric will be posted on Canvas.

Note: Forgetting to submit your work, 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).

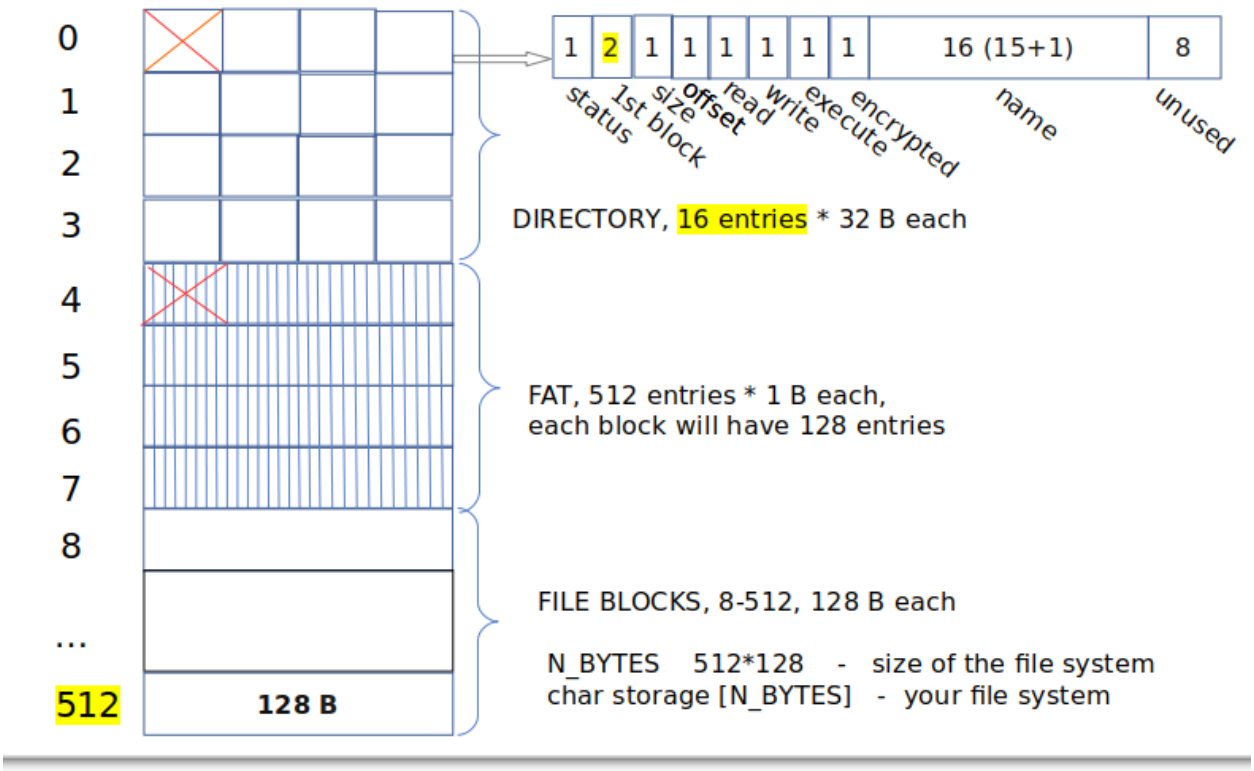
You may discuss the project requirements and the concepts with me or with anyone in the class.

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.)

Btrfs File System Scheme on Next Page


0
1
2
3
4
5
6
7
8
...
512
128 B
1
2
1
1
1
1
1
1
16 (15+1)
8
status
1st block
size
offset
read
write
execute
encrypted
name
unused
DIRECTORY, 16 entries * 32 B each
FAT, 512 entries * 1 B each,
each block will have 128 entries
FILE BLOCKS, 8-512, 128 B each
N_BYTES 512*128 - size of the file system
char storage [N_BYTES] - your file system