CSE 3320 Operating Systems
Fall 2021, © DL, UTA, 2021

Programming Assignment 3
File System Internals
Due: On Canvas

Description:
You will develop and implement a small file system ("FS"). It is similar to some of the basics of Unix as well as CP/M file systems.
Your file system will not be part of an operating system, but, similar to most modern file systems, it will run in several different operating systems to provide a "portable" file system.

Details:
Your FS will use a file (for example "disk01"), rather than directly using a physical flash or disk, to store data.

You may have several disk-like files (for example: disk01, disk02), used to store data.

The data stored in disk01 may be a user's programs, text files, other data files, or any type of binary information. In addition to the data stored, your FS will need to store other, meta-information, such as free space (blocks), directory details, and possibly other information. The FS directory is flat (one level) fixed sized, has a user name associated with each file, and has fixed sized "blocks" (entries).
You should use fixed size blocks (similar to disk blocks) of size 256 (or 1KB, your choice, examples based on 256) bytes to store files and all meta-data in your "disk".
(Your "disk" (for example "disk01" is logically divided into a number of "sectors", which are fixed size blocks. Everything that is stored (persistent) is in these blocks)

Your program (the "FS" executable) should provide the following operations:
(These operations deal with the "disk", not individual user files)
Createfs #ofblocks – creates a filesystem (disk) with #ofblocks size, each 256 bytes
For example Createfs 250 creates a "virtual disk", a file that will be initialized to 250 blocks of 256 bytes each. It is created in memory, and initialized.

Formatfs #filenames #DABPTentries
For example Formatfs 64 48 reserves space for 64 file names and 48 file meta data, Note that some file names may "point" to the same file metadata, in this example there can only be 48 unique files.

Savefs name– save the "disk" image in a file "name"
Openfs name- use an existing disk image
For example Savefs disk01 or Openfs disk01
These commands same the memory "image" (contents) to an external file, in this example, it is called disk01, but can be called anything, the openfs command retrieves the image/contents from the file and puts into memory.

List – list files (and other meta-information) in a FS
List what is in "your" directory

Remove name –remove named file from fs
Delete a user file, should reclaim the directory entry and file sectors

Rename oldname newname – rename a file stored in the FS
Just change user file name

Put ExternalFile – put (store) Host OS file into the disk
Get ExternalFile – get disk file, copy from "disk" to host OS file system
These operations put and get a user file from "outside" to and from your file system

User name – show/change name of user who owns this file
Link/Unlink – Unix style file linking
These are some more, common, meta operations, only changes something in directory, not the data file contents

Bonus: Set/Use file permissions for r/w/x, implement subdirectories, "check disk"
Implement in either the "Go" or "Rust" programming language (20 to 75 point bonus)

Implementation:

(Note: these names and acronyms are hints, there are other methods and data structures that may also work.)

Your FS should have 4 (or more, if easier to implement) sections:
A FileNameTable (FNT), a directory and a disk attribute/block pointer table (DABPT), and the data blocks.

The FNT should be of size allocated, each entry should contain a 50 char (maximum) file name and an inode pointer (index to DABPT)(blocks).

The DABPT should be allocated from disk blocks, 4 entries per block, where each entry should contain a file meta-information (FileSize, last time+date (secs), pointers to data blocks), user name

The Block Pointer Table has direct pointers to data blocks, and one additional pointer to another entry in the Table, if needed (for big files), these may be chained for very large files. (Similar to CP/M extents)

Since disks (and some meta-information) are fixed size, many small or one large file might not fit on the "disk". File names, file attributes and other file information stored in FS are restrictive (for example, file creation time).