Assigment 3 - DV1629

DVGHI20 – Lukas Einler Larsson & Kushtrim Qollakaj

Tasks

All functions have error checking before performing their intended action.

Task 1. Implementation of basic file operations

FS::format

Formatting the filesystem involves setting all FAT entries as FAT free expect the first two disk blocks where the first block is the root block and second one is the FAT table itself. The present working dictionary is set to root ("/") and the root directory is cleaned by putting files to size 0.

FS::create

Creating a file in the filesystem is performed by first error checking, then receiving input from the user to the content of the file, then the file receives the necessary free blocks on the disk and is written to the disk, lastly the directory the file exist in is updated.

FS::cat

If the user Cat a file there are first error checking if the file e.g., is a directory, then the filesystem gathers information from the file wanted (the blocks used to write the data etc) and prints it out the data of the file in the terminal.

FS::ls

Ls lists all the files in the present working directory. Including the files' name, file type, accessrights and size of the file

Task 2. Implementation of file operations, part 2

FS::cp

This function copies the data of the first argument to a new file and pastes the data into the new file (second argument of command). The second argument can either be in the present directory or in another directory in the filesystem. This action is performed by reading the disk blocks of the sourcefile and then writing the same data to the destination file that has been given free blocks to write on.

FS::mv

Mv either renames the file given in the first argument to the filename given in the second argument. If the second argument is a directory the file is being moved to the directory given in the second argument. The action of moving a file into another directory is performed by copying the data of the file into the given directory and then updating the source and destination directory.

FS::rm

Removing a file in the filesystem involves first finding the file and then marking the file in the directory as free. Also, the disk blocks used by the file are marked as FAT_FREE.

FS::append
Appending one file to another first starts with lots of error checking, then reading the data from the sourcefile and then reading the data from the destinationfile and then adding the data together then rewriting the combined data to the destinationfile.

Task 3. Implementation of directory hierarchies
FS::mkdir
Mkdir is similar to FS::create but doesn't read any content data from the user to the file. The file created is instead of the file type TYPE_DIR and receives one freeblock to hold directory entries in. There can be 64 directory entries in each directory. A directory created with mkdir always consists of one entry in the directory right away. The ".." directory that allows for the user to move back in the hierarchy via the FS::cd command.

FS::cd
This command allows the user to move around in the file hierarchy, cd = Change Directory.

FS::pwd
This command prints out which Present Working Directory the user is in.

Task 4. Implementation of absolute and relative paths
This implementation was challenging because the pwd needed to correspond to the pwd disk block. If these two don't correspond they can easily confuse the user and so they must always work in parallel.

Task 5. Implementation of access rights
FS::chmod
This command was implemented by first reading the access right given in the first argument and then finding the file given in the second argument. The file given is then updated with the access right.

Help functions
gather_info_new_dir_entry
This function is called in the FS::create function and allows the filesystem to gather information from the user input when creating a new file.

gather_info_old_dir_entry
This function gathers info about an entry that already exists in the disk block given as an argument.

check_if_file_in_dir
This function checks if a file is found in the directory. The function returns -1 if the file is found and it is a TYPE_FILE, if the file is found in the directory already but is a TYPE_DIR the return value is -2. If the file is not found in the directory the return value is 0.

save_entry_on_disk
This function is used in filesystem commands such as FS::append, FS::create etc and allows the filesystem to find freeblocks to write to and then writes the given data in those blocks. The directory entry first_blk value is also updated.

save_entry_in_dir
This function is used to update the directory with the entry given in the argument.

get_filename
This function fetches the filename from a given filepath.

privilege_string
This function transforms the integer value of the files access rights to a string.

get_dir_blocks
This function's argument is a filepath. The function traverses the filepath given and gathers all the directory blocks that the filepath has. It then returns all the directory blocks found in a vector. If a directory isn't found in the given filepath the vector consists of an integer -1.

check_if_dir_full
This function is used to check if a directory is full.

destination_dir_check
This function returns a 0 if the filepath given consists of directorys and a -1 if the filepath is just a filename.

privilege_check
This function uses an ANDing operation to see if the access rights to a file allows for reading/writing/executing.