CS 1550

Week 13 Project 4- Part III

TA: Pratik Musale

getattr

- if (path is the root (“/”))
    - return that it is a directory with given permissions in cs1550.c and with 2 links
- parse the path using sscanf
- Use the return value of sscanf to determine if the path is a subdirectory (Case 1), a file without an extension (Case 2), or a file with an extension (Case 3)
- Case 1: if(path is a subdirectory (e.g., “/dir1”))
    - read the root directory block into a variable of type cs1550_root_directory
    - iterate over the array of directory entries using the number of directories as the number of iterations
        - Each iteration compares the directory name from the path with the directory name in the entry using strncmp
        - if directory found
            - return that it is a directory with given permissions in cs1550.c and with 2 links
        - else return not found error
- Case 2 and 3: if(path is a subdirectory and a file without orweith an extension (e.g., “/dir1/file1”))
    - read the root directory block into a variable of type cs1550_root_directory
    - iterate over the array of directory entries using the number of directories as the number of iterations
        - Each iteration compares the directory name from the path with the directory name in the entry using strncmp
        - if directory not found
            - return not found error
        - if directory found
            - get the block number from the entry
        - read the file directory block into a variable of type cs1550_directory_entry
    - iterate over the array of file entries using the number of file as the number of iterations
        - Each iteration compares the file name (and the extension in Case 3) from the path with the file name (and extension n Case 3) in the entry using strncmp
        - if file found
            - return that it is a file with given permissions in cs1550.c, with 1 link, and with the correct size from the file entry
        - else return not found error

mkdir

- parse and validate the path
  - check getattr for more details on how to parse and validate the the directory doesn’t exist
- read the root directory block into a variable of type cs1550_root_directory
- check if maximum number of directories already hit
  - if so, return no space error
- copy the directory name from the path to the directory name at entry at index num_directories
- Use the last_allocated_block as the block number for the new directory and increment last_allocated_block
- Increment num_directories
- seek to the beginning of the file and write the root directory back to the file
- return success

readdir

- parse and validate the path
  - make sure that it is a directory path (the root “/” or a subdirectory “/dir1”)
  - if not return permission error
- read the root directory block into a variable of type cs1550_root_directory
- Use the filler function to fill the provided buffer with the two links “.” and “..”
- Case 1: (path is the root)
  - Iterate over the entries ion the root directory and use the filler function to insert the directory name into the provided buffer
- Case 2: (path is a file directory)
  - Find the directory entry and retrieve the block number
    - if not found, return not found error
  - Seek to and read the file directory block into a variable of type cs1550_directory_entry
  - Iterate over the entries ion the file directory and use the filler function to insert the file name and extension (if it exists) into the provided buffer

mknod

- parse and validate the path
  - check getattr for more details on how to parse and validate the the directory exists and the file doesn’t exist
- read the root directory block into a variable of type cs1550_root_directory
- iterate over the array of directory entries using the number of directories as the number of iterations
  - Each iteration compares the directory name from the path with the directory name in the entry using strncmp
  - if directory not found
    - return not found error
- if directory found
  - get the block number from the entry
- read the file directory block into a variable of type cs1550_directory_entry
- check if maximum number of files already hit
  - if so, return no space error
- copy the file name (and extension if exists) from the path to the file name at entry at index num_files
- Use the last_allocated_block from the root block as the block number for the index block and increment last_allocated_block
- Seek to and read the index block into a variable of type struct cs1550_index_block. Write the value of the last_allocated_block as the first entry in the index block array. Increment last_allocated_block for the first data block of the file
- Increment num_files
- seek to the beginning of the file and write the root directory back to the file
- seek to the file directory block location and write the file directory back to the file
- seek to the index block location and write the index block back to the file
- return success

read

- parse and validate that the path contains a directory and a file name
- read the root directory block into a variable of type cs1550_root_directory
- iterate over the array of directory entries using the number of directories as the number of iterations
    - Each iteration compares the directory name from the path with the directory name in the entry using strncmp
    - if directory not found
        - return not found error
    - if directory found
        - get the block number from the entry
    - read the file directory block into a variable of type cs1550_directory_entry
- iterate over the array of file entries using the number of file as the number of iterations
    - Each iteration compares the file name (and the extension in Case 3) from the path with the file name (and extension n Case 3) in the entry using strncmp
    - if file not found
        - return not found error
    - if file found
        - read the index block
        - Use the offset parameter to determine the index inside the array of data blocks inside the index block (offset / block size)
        - read the data block
        - Use the offset parameter to determine the index inside the data block (offset % block size)
        - User memcpy to copy size number of bytes from the data block starting at (offset % size) into the provided buffer
        - if you need to read the next data block(s), retrieve the block number(s) from the index block

Hints

- Before parsing the path using sscanf make sure that the length of the directory name, file name, and extension is within the maximum; otherwise, you may get segmentation faults
  - you can iterate character by character on the path string and count the number of characters between the slashes (for the directory name) and between the slash to the dot (for file name), and from the dot to the end of the string (for the extension).
- Before reading or writing a block, make sure that you seek (using fseek) into the block
  - fseek takes the offest in terms of number of bytes not in terms of block number; so, you have to multiple the block number by the block size.
  - Use SEEK_SET to seek from the beginning of the file