CS416 – Assignment 3

Names: Nicholas Prezioso, Benjamin Cahnbley, and Marcella Alvarez

Username of iLab: njp207, bc499, and ma1143

iLab Server: flyweight

Design:

There is a set file size of 16MB, therefore more than 16MB cannot be written to in our file system. This 16 MB is broken into chunks of 512. Each chunk represents a block of memory, either as an inode (block), or a data block (i.e. data for a file, denoted as data_block). Inodes (blocks) represent both empty spaces, directories and files, while data blocks are inserted in place of an inode, hold the buffer/data of the file, and are only created in sfs_write.

Structs:

Every inode/block struct keeps track of its:

Int type [-1, error; 0, directory; and 1 file]

struct stat s [st_uid, st_gid, mode, etc.]

int p[BlockArraySize] [Array of links, if a directory]

char path[255] [Given Name/Path]

Every data_block struct keeps track of its:

Int type [-1, error; 2, data]

Int size [Amount of data used]

char data[512-(2*sizeof(int))] [The data itself]

Methods:

int getBlock(const char* path)

Traverses through the data blocks, while simultaneously removing the slashes of the given const char* path, until the desired directory is reached and/or file is found. Returns -1 upon failure to find file/directory, else returns the block number in which the path name resides in.

void *sfs_init(struct fuse_conn_info *conn)

Initializes the filesystem by open the diskfile and initializing all blocks/inodes

void sfs_destroy(void *userdata)

Called when the filesystem exits. Returns all blocks to their original values in init (0/Null)

int sfs_create(const char *path, mode_t mode, struct fuse_file_info *fi)

Checks to see if a file exists, then if not, creates it with the specified mode by finding an open block, and opens it. Returns -1 on failure (e.g. not enough room in filesystem, or file already exists in that directory).

int sfs_unlink(const char *path)

Removes (deletes) the given file, symbolic link, hard link, or special node (basically anything that relates to this file).

int sfs_getattr(const char *path, struct stat *statbuf)

Fills the elements of the “stat” structure for the given const char *path. Returns -1 upon failure (i.e. file does not exist)

int sfs_open(const char *path, struct fuse_file_info *fi)

Checks to see if the file exists. If so, it checks to make sure the flags are valid. Opens the file. Returns -1 upon failure (e.g. flags invalid or file does not exist)

int sfs_release(const char *path, struct fuse_file_info *fi)

Checks to see if a file exists. Returns -1 on failure (e.g. file does not exist, or type is not a file)

int sfs_read(const char *path, char *buf, size_t size, off_t offset, struct fuse_file_info *fi)

Read bytes from the given file into the buffer, beginning offset bytes into the file.

int sfs_write(const char *path, const char *buf, size_t size, off_t offset, struct fuse_file_info *fi)

Write bytes from the given buffer into a file.

Returns -1 on failure (e.g. if the offset for a given block to write was not the same as the buffer, the file does not exist, or the block is not a file)

int sfs_readdir(const char *path, void *buf, fuse_fill_dir_t filler, off_t offset, struct fuse_file_info *fi)

Returns one or more directory entries (struct dirent) to the caller.

int sfs_opendir(const char *path, struct fuse_file_info *fi)

Checks to see if the directory exists. If so, it checks to make sure the flags are valid. Opens the directory. Returns -1 upon failure (e.g. flags invalid or directory does not exist).

int sfs_releasedir(const char *path, struct fuse_file_info *fi)

Checks to see if a file exists. Returns -1 on failure (e.g. file does not exist, or type is not a file)

int sfs_rmdir(const char *path)

Removes the given directory. Returns -1 upon failure (e.g. invalid path, not a directory, or isn't empty)

int sfs_mkdir(const char *path, mode_t mode)

Creates a directory with the given name using the given permissions denoted by mode.