CS 179F

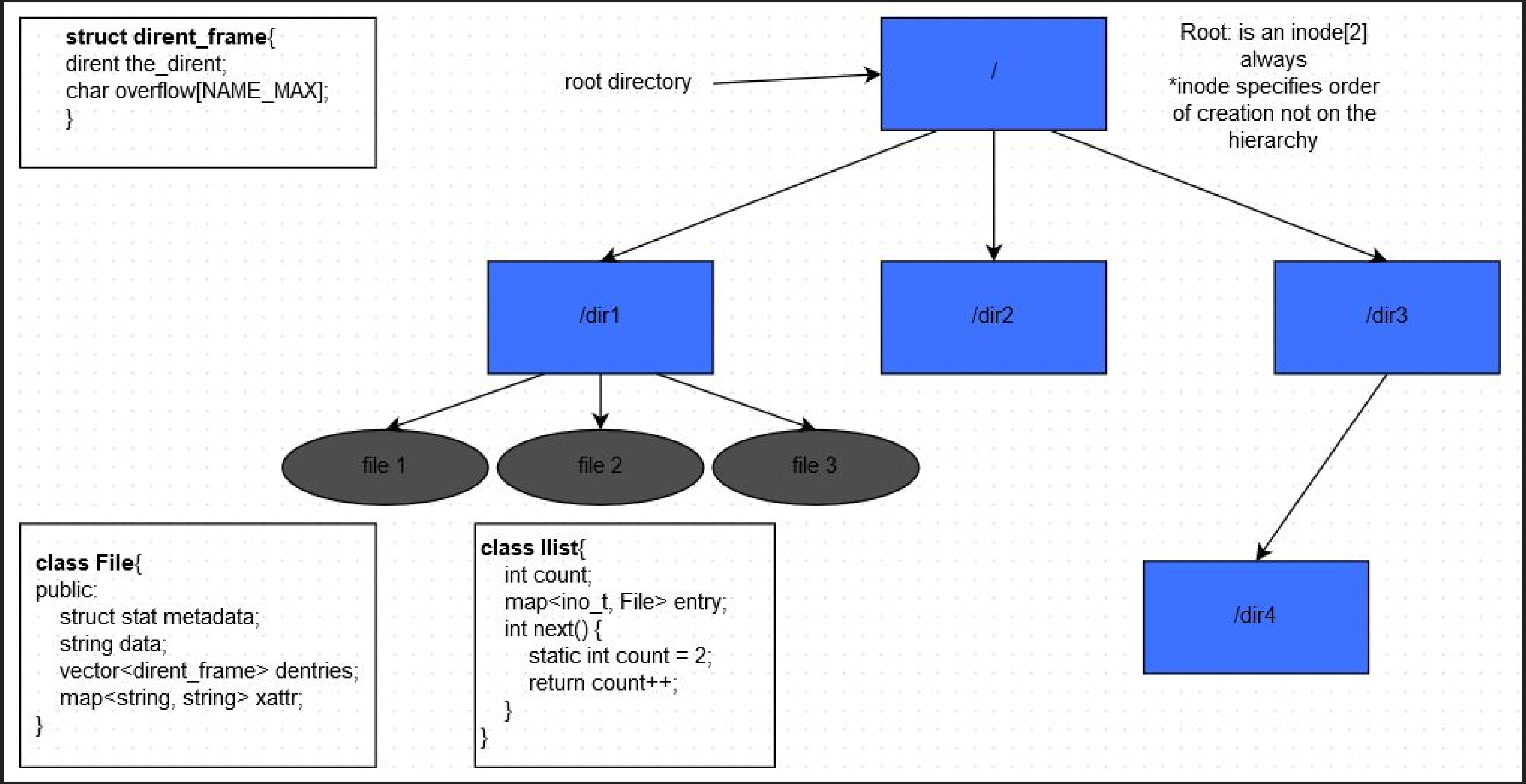
Christopher Kotyluk, Marcel Tawamba, Selik Samai,
Jesus Reyes, Gregory Beatty-Fechter

https://github.com/ckotyluk/CS179F_BBFS_Project

Project Overview

- Initially meant to mirror BBFS (Big Brother File System)
- FUSE served as an interface to execute our system calls
- Due to failures to use FUSE to properly exploit the system calls, we transitioned to testing the system calls using our own framework
- Overall we have 21 system calls implemented
    - Including symbolic links and extended attributes

Filesystem Structure


root directory
/
Root: is an inode[2] always
*inode specifies order of creation not on the hierarchy
/dir1
/dir2
/dir3
file 1
file 2
file 3
/dir4

```
struct dirent_frame{
dirent the_dirent;
char overflow[NAME_MAX];
}
```

```
class File{
public:
    struct stat metadata;
    string data;
    vector<dirent_frame> dentries;
    map<string, string> xattr;
}
```

```
class llist{
    int count;
    map<ino_t, File> entry;
    int next() {
        static int count = 2;
        return count++;
    }
}
```

System Calls Walkthrough

High Priority System Calls

Function: mknod()

Creates a file

- Used to create regular files (creat)
- Used to make directory files (mkdir)

Error Handling:

- Checks that the path is valid

Design:

- Create dirent and add to parent

Testing Status: **Heavily Tested**

Function: creat()

- Creates a new file or rewrites an existing one.

Error Handling:

- Checks that the path is valid
- Checks that permissions are valid

Design:

- Calls mknod to create the file

Testing Status: **Heavily Tested**

- Create files in current directory
- Create files via path
- Creates `.` file

Function: rename()

- Renames a file, moving it between directories if required

Error Handling:

- Checks for valid paths
- Does not allow overwriting of existing files

Design:

- Similar to design of link

Testing Status: **Heavily Tested**

- Renaming file in curr_dir works
- Can rename `.` and `..` dir when root

Function: link()

- Creates a new link (also known as a hard link) to an existing file.

Error Handling:

- If newpath exists, it will not be overwritten.
- Oldpath must have a valid path
- Oldpath must be a file

Design:

- Add directory entry to parent of newpath

Testing Status: **Heavily Tested**

- Can link files in current directory
- Can link files via paths

Function: unlink()

- Deletes a name from the file system.
- The file is removed once there are no more links to it.

Error Handling:

- Check that path is valid

Design:

- Delete directory entry
- Decrement link count; delete file if 0

Testing Status: **Heavily Tested**

- Can unlink ‘.’ or ‘..’

Function: open()

- Opens existing file and creates if non-existing.

Error Handling:

- Verify valid inode

Design:

-

Testing Status: **Heavily Tested**

Function: close()

- Decreases the nlink or count of links to this file.

Error Handling:

- Checks that the file exists

Design:

- Decrements the number of links to this file
- Erases the file when the number of links is 0

Testing Status: **Lightly Tested**

Function: pread()

- Reads n characters from a file starting at

Error Handling:

- Handles requests for more characters than exists

Design:

- data is a string variable
- requests starting position and number of characters to read

Testing Status: **Lightly Tested**

- Can write to files
- Able to write to directories

Function: pwrite()

- Writes bytes from the buffer to the file descriptor.

Error Handling:

- Checks of writing to regular file

Design:

- data is a string variable
- Requests starting position and input data

Testing Status: **Lightly Tested**

- Can read from files that have been written to
- Able to read from directories

Medium Priority Functions

Function: access()

- Checks real user's permissions for a file.

Error Handling:

- Checks if filepath is valid
- Checks that the permissions in mask match the user permissions of file

Design:

-

Testing Status: **Lightly Tested**

-

Function: chown()

- Changes ownership of file.

Error Handling:

- Checks nonexistent paths

Design:

- If user id doesn’t exist, number is outputted instead

Testing Status: **Lightly Tested**

- Changes ownership of file in current directory and via path

Function: chmod()

- Changes permission of a file.

Error Handling:

- Checks of input file is directory or file
- Handles bad file handles

Design:

-

Testing Status: **Lightly Tested**

- Change permission of dir in current directory
- Change permission of file via path
- Can change directories to files, including `. ` and `..`

Low Priority Functions

Function: symlink()

- Creates a soft link that makes one file point to another file

Error Handling:

- Cannot overwrite “.” and “..”
- Link already exists

Design:

- The linked path is written into the data of the file
- The file is marked as a symbolic link using chmod

Testing Status: **Lightly Tested**

Function: readlink()

- Read the link path from the symbolic link file.

Error Handling:

- Check that the file exists
- Checks that the file is a link

Design:

- Modified find_ino so that it handles symbolic links
- Implemented find_real_ino to ignore symbolic links

Testing Status: **Lightly Tested**

Function: utime()

- Changes the access time and modify time of a file.

Error Handling:

- Checks that the time struct exists
- Checks that the file exists

Design:

- Set the values of the time struct to the time values in the file’s metadata

Testing Status: **Lightly Tested**

Function: truncate()

- Truncates the files based on the based in size
- Strinks the file if the new size is smaller than the current file size
- Expands the file with null characters if the new size is larger than the current file size

Error Handling:

- Checks that newsize is a valid size
- Checks that the file exists
- Checks that it is not resizing a directory

Design:

- Gets the inode of the path, then calls ftruncate

Testing Status: **Lightly Tested**

Function: ftruncate()

- Truncates a file to a specified length. The difference with truncate is that it takes a file descriptor instead of a **const char*** as first parameter.

Error Handling:

- Checks that newsize is a valid size
- Checks that the file exists
- Checks that it is not resizing a director

Design:

- Uses the string resize() function to change the length of the data string

Testing Status: **Lightly Tested**

Function: statvfs()

- Gets the information about the file system

Error Handling:

- Checks that the file exists
- Checks that the statvfs struct exists

Design:

- Only focuses on the number of inodes and the max name size
- Sets other values to 0 since they are not relevant to our file system

Testing Status: **Lightly Tested**

Very Low Priority Functions

Extended Attributes

- Adds extended metadata to a file
- These attributes are not needed nor defined by the filesystem
- Can be used to store a file’s author, access control lists, other permissions, etc
- Stored as a name and value pair
- Functions are used to interact with these attributes
- Modified File struct so it contains a map that stores the name and the value as strings

Function: lsetxattr()

- Sets the value of an attribute based on the given name and given value

- The `l` means that the attribute is added to the file, not to the linked file

Error Handling:

- Check that the path is valid

Design:

- Inserts the pair <name, value> onto the file
- Ignores the flags which determines whether to create or replace an attribute

Testing Status: Lightly Tested

Function: lgetxattr()

- Gets the value of an attribute based on the given name

- The `l` means that the attribute is read from the file, not the linked file

Error Handling:

- Checks that the path is valid
- Checks that the attribute exists

Design:

- Accesses the attribute map of the file
- Find the value based on the name
- Stores the value in a passed in buffer

Testing Status: **Lightly Tested**

Function: lremovexattr()

- Removes the attribute with the corresponding name

- The `l` means that the attribute is read from the file, not the linked file

Error Handling:

- Checks that the path is valid
- Checks that the attribute exists

Design:

- Accesses the attribute map of the file
- Removes the attribute with the corresponding name

Testing Status: **Lightly Tested**

Function: llistxattr()

- Outputs a list of all values for the attributes of a file

- The `l` means that the attribute is read from the file, not the linked file

Error Handling:

- Checks that the path is valid
- Checks that the attribute exists

Design:

- Accesses every attribute of the file
- Writes the values into a string with the values separated by null characters

Testing Status: **Lightly Tested**

Major Bugs

- Cin corruption (creat, mkdir, chown, chmod)
- Handling “.” and “..” directories
- Reading and writing to and from directories
- Changing directories into files with chmod

Possible Future Work

- Linux error codes
- persistence of files via block allocation of strings, maps, and vectors
- concurrency using multithreading.
- Integration with FUSE, possibly using BBFS as an adaptor.
- Improving performance by replacing STL data types such as maps and vectors with arrays and free-entry lists, stacks, or bitmaps.
- Improving reliability via the use of checksums and redundancy
- Sparse (aka gapped) allocation of regular files, as seen in Unix. (C++ strings do not currently support gaps.)
- Making provisions for large (multi-terabyte) files.