**CS551 Project 3: File System Tools Implementation**

**Design Document**

**Team Members**

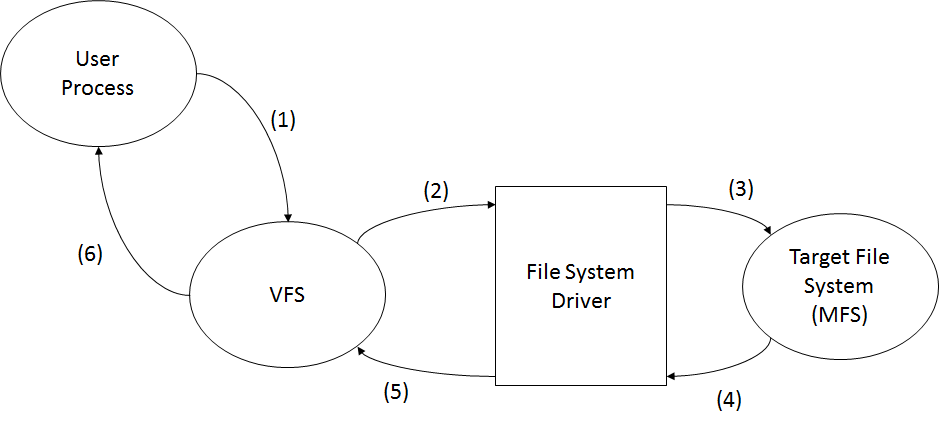
Balaji A R (A20347964)

Paulo Vitor Becker Ayrosa Monterio de Andrade (A20341532)

Sai Ravali Nannuru (A2035434)

**System Calls Organization:**

For this particular set of system calls a series of messages are exchanged between the Virtual File System Server (VFS), the File System Driver Library (libfsdriver) and the system which ultimately has access to the actual file system’s constructions, which is in this case the Minix File System (MFS).



(1) – The user process makes a request to the Virtual File System by using a System Call to that server, using the corresponding system call name;

(2) – The Virtual File System receives the request and translates this syscall number into a do\_syscall(args) using the appropriate tables in its source code mapping this syscall to a request\_syscall() to the File System Driver;

(3) – The File System Driver serves as an abstraction to which file system is being used by the Operating System; it transmits the request message from VFS to the target file system (in this case, Minix File System), which in turn maps this request into a function to deal with the request;

(4), (5) & (6) – The request gets processed and returns through these messages, with each of the requests which were waiting as activation records receiving its corresponding return value.

The exception generating system calls are mostly residing outside of the file system tools developed for this project; the calls to VFS, FS Driver and MFS can generate exceptions on a series of situations:

* Path doesn’t exist,
* I-node allocation failed (multiple reasons),
* Zone allocation failed (multiple reasons),
* Superblock fetch failed,
* Superblock get failed, etc.

**User Manual:**

1. int inodewalker()

Description:

This system call will run through all of the i-nodes in the system, check which one is assigned and print out the total number of allocated i-nodes in the Operating System’s file system.

The target device is the currently opened one, described in the variable dev.

Exception:

Bitmap allocation fail:

1. int zonewalker()

Description:

This system call will run through all of the zones in the system, checking which one is assigned and print out the total number of allocated zones in the Operating System’s file system.

Exception:

None generated by this particular syscalls, just the internal syscalls to VFS and their subsequent calls.

1. int zinfo(struct stat path\_stat)

Description:

This system call will run from a root directory specified as the path\_stat variable and list all the i-nodes associated with this path and its children.

This system call is used inside a directory walking function performed as an option in the test program.

Exception:

None generated by this particular syscalls, just the internal syscalls to VFS and their subsequent calls.

**Disk recovery tools:**

There are common scenarios when a inode of a file or a inode of a directories are corrupted except when it happens inside directory to a file system like with journal in file system meaning, system will boot normally as it should but after sometime, the partitions will be remounted as Read-Only file system since kernel is unable to read the blocks in the journal, the only way to fix this issue by rebuilding the journal. Apart from that it generally throws errors like:

"Corrupt inode (1): Input/output error

Failed to open inode $MFTMirr: Input/output error"

"Failed to open inode $MFTMirr: Input/output error"

In the above conditions, in core inodes can considered bad if all paths to a device have been disabled.  If the paths are re-enabled, and the file system is still enabled, it is then possible for these in core inodes to be flushed to disk and the superblock marked as needing a full fsck.  A subsequent full fsck will clear these inodes, deleting the file.  The most relevant thing to do for inodes considered bad due to read failures, as the inode was not being updated at the time, and has the most probability of being recovered successfully.

After a link failure has been detected, the file should be analyzed for possible inode failures.  File System will print messages to files and will indicate which inodes are marked bad.

Unmount the file system in order to attempt repairs on corrupted inodes.  The superblock can be analyzed on the failing file system to verify that it has been marked as needing a full fsck (seen in the above question). Once we know the file system has corruption in it, we need to perform a full backup of data and dump the metadata (Saving the metadata is a good idea in case there are problems with fsdb )

Run a full "fsck -n" to see which inodes are bad.

Allocation flags have to be set to 0x0 using fsdb.  This procedure involves writing permissions to file system structure. The inode allocation flags has to be cleared.  Then repeat with verifying with fsck. Run a full "fsck -y" which is safe. Once it's safe Mount the file system and check the inodes.

A block (or blocks) allocated to a file is damaged:

As remaining other damage recoveries, a bad block has to be detected first and then they have to be fix the file system. Bad block numbers in an inode might be caused by an indirect block not being written to the file system.

The fsck program checks each block number claimed by an inode to see that its value is higher than that of the first data block and lower than that of the last data block in the file system. If the block number is outside this range, it is considered a bad block number. The fsck program prompts to clear the inode.

Whole disk scan: badblocks -v /dec/sdx

Single partition scan : badblocks -v /dev/sdxy

x,y are drive letter and partition number.

From these block numbers of bad blocks and their partition numbers are analysed.

If bad blocks didn't run using the block size of the file system is using then convert block numberto match it.

(original block number) / ((file system block size) /(bad blocks block size))

d blocknumber 0 i

d command for display, the block number, the offset (set to 0), and the display format i for inode and from this we get inode number

Now we need to force the disk to reallocate the bad block with smartctl command.

**Test Cases:**

1. Request an i-node listing using the option #1 in the test program. Returned value must be 0 (user process gets this but does nothing) and the used i-node list is print out, followed by the number of used i-nodes and the total number of i-nodes in the Operating System’s file system.
2. Request a zone listing using the option #2 in the test program. Returned value must be 1 (user process gets this but does nothing with it) and the used zone list is print out, followed by the number of used zones and the total number of zones in the Operating System’s file system.
3. Request a directory walk listing associated i-nodes using option #3 from the test program with a valid path when prompted, such as /home that also isn’t too long, otherwise it will take a long time to scan and list all of the children. All of the i-nodes associated with the chosen path must be listed, as well as the directories.
4. Request a directory walk listing associated i-nodes using option #3 from the test program, entering an invalid path when prompted (an inexistent path in the file system). The program should notice the incorrect input value, print a message warning the user of such mistake and re-run the option list from the test program.
5. Request a directory walk listing associated zones using option #4 from the test program, entering a valid path (such as /home) when prompted. The program should list all the associated inodes for the chosen path and list the subdirectories.

Also, the chosen path should’t be too “deep”, as it would take a long time to print out all of the files to the terminal.

1. Request a directory walk listing associated zones using option #4 from the test program, entering an invalid path when prompted (an inexistent path in the file system). The program should notice the incorrect input value, print a warning message to the user of such mistake and re-run the option list from the test program.