Jacklyn McAninch A20436746
Khiem Truong A20437990

Design

Both of these walkers were implemented in walkers.c, but they take different approaches for essentially performing the same function. For **directoryWalker**, the list of dirents was retrieved from reading information from disk in order to get the name of the file. Inode number and information is extracted through this file name in the directoryWalker method, using the built in method dirlookup() from xv6. This function is invoked by the syscall walkdir(). In contrast, the **inodeTBWalker** instead goes through inodes by their inode number sequentially, rather than by directory. This is done by directly reading the allocated data block associated with the inode number, instead of consulting the list of dirents in a specified directory. This is invoked by the walkInodes() syscall.

For comparing both of these walk methods, the inodes allocated for each method are stored in its own array. A 1 at index i signifies an inode with inum i has been allocated and discovered by the function. Inside the **compareWalker** method, invoked by syscall compareWalkers(), these two arrays are compared to see if the output was the same (expected output for an undamaged file system is yes). A third array, comp[], is used to flag any differences between the two original arrays, where a 1 indicates that one walker found an allocated inode that did not appear allocated in the other walker. This comp[] array is used later for fixing damages to the file system.

To intentionally damage the file system, the user uses the eraseInfo() syscall, which calls the **eraseInf** function inside walkers.c. This one in turn calls the method erase() inside fs.c (so that it can reach some static inode methods). It damages the specified directory by loading it and calling itrunc(), which sets the size of this directory to zero and removes all its pointers to the block locations of the directory’s files. This call also resets the array for keeping track of allocated inodes as discovered by directoryWalker.

Lastly, to restore the file system, the user must use the fixdamagedFS() syscall, which invokes the **fixDmgFS** function in walkers.c. Similarly to eraseInf, this was also implemented inside fs.c to access static inode methods, under the function name fix(). To fix the damaged file system it first calls compareWalker to obtain the differences between the two inode allocation arrays. It then counts the number of damaged files and locates the directory that was wiped (a directory of size 0). If the number of damaged files is greater than the number of files that can fit in one directory, the operation cannot proceed because all the files must be restored to the same folder. After locating the target directory, all other inodes which are flagged by comp[] are linked back to this directory and the damaged directory's address blocks are updated to reflected these. Finally, the restored inodes must then be linked in the dirent structure as well so their names reflect the inode again. **Bonus:** with this solution, the inodes were restored by loading the on-disk data structure, but if that was also damaged, you could access the log which keeps track of all inode activity and restore the damaged inodes and directories from there.

Files changed

- created file walkers.c to define methods for walking the inodes/directories
- edited fs.c to enable erase() and fix() methods to call static methods in fs.c
- edited defs.h to include new methods
- edited syscall.c, syscall.h, sysfile.c, usys.S, and user.h to implement new syscalls
- created file test1.c to test the methods in walkers.c

Syscall manual pages

walkDir:

- Descriptions: the walkDir() syscall will print the names of each file and directory (recursively travel into any child directories as well) in a file system tree, given a starting point. It will also print out the inode number associated with the file or directory. After all directories and files names with their corresponding inode number is printed, the syscall returns -1 if the path is invalid or error reading file, else returns 0
- Arguments: char *path
- Errors: walkDir() will run into error if the file can't be read, the path is invalid, or the directory is bad

walkInodes

- Descriptions: the walkInodes syscall goes through the inode table and prints out all of the allocated inode along with its type and size. The syscall returns 0 after finished printing the allocated inodes. It has no error case.
- Arguments: none

compareWalkers

- Description: the compareWalkers syscall will compare the 2 walkers array and print out the status of all inodes, if it is in both of the walkers, or if it is missing from one of the walkers. After it finished with printing out the comparison between walkers, the syscall returns 0
- Arguments: none
- Error: Syscall will return -1 if either the directory array or inode array is empty

eraseInfo

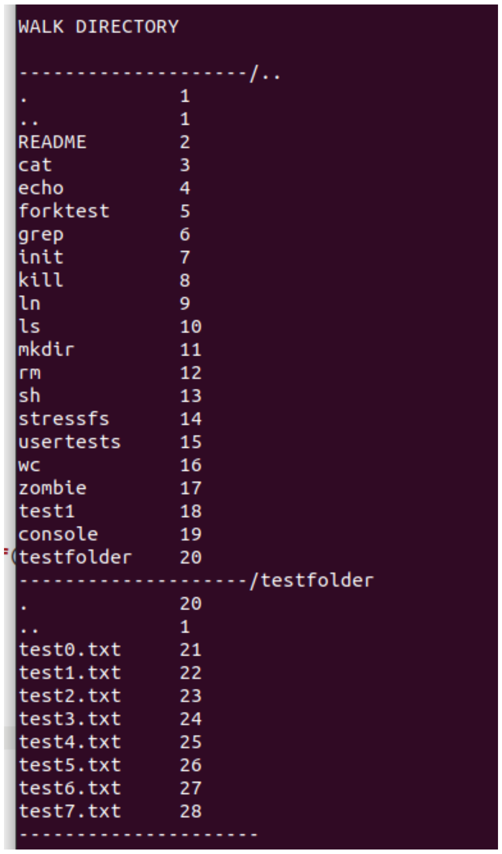
- Description: when invoked by a user program, will call the method eraseInf() from walkers.c. From there, in order to access private methods inside fs.c, it will call the method erase() in fs.c. This one checks that the specified inode is a directory, and if so calls the xv6 function itrunc() to erase all data from the inode in memory and remove them from the cache. This also erases the address pointers of the on-disk representation of the inode.
- Arguments: int inode (must be a directory)
- Error: none

fixDamagedFS

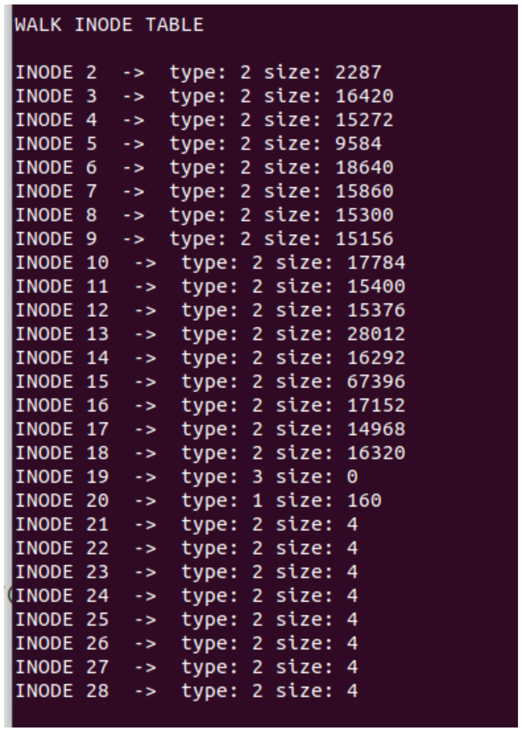
- Description: this system call locates the damaged directory and then restores the on-disk representation of damaged inodes to links back to the directory. This function also calls compareWalkers from inside of it.
- Arguments: none
- Error: if reading the dirent structure from disk fails, the function will return -1 and exit, otherwise on successful run returns 0

Test cases

The file system we used consisted of the xv6-provided files plus one extra directory which contains 8 sample files, each containing 4 bytes each. This is good test data it allows us to erase quite a bit of information and restore it back to the directory it came from. Here is the process as executed on xv6:

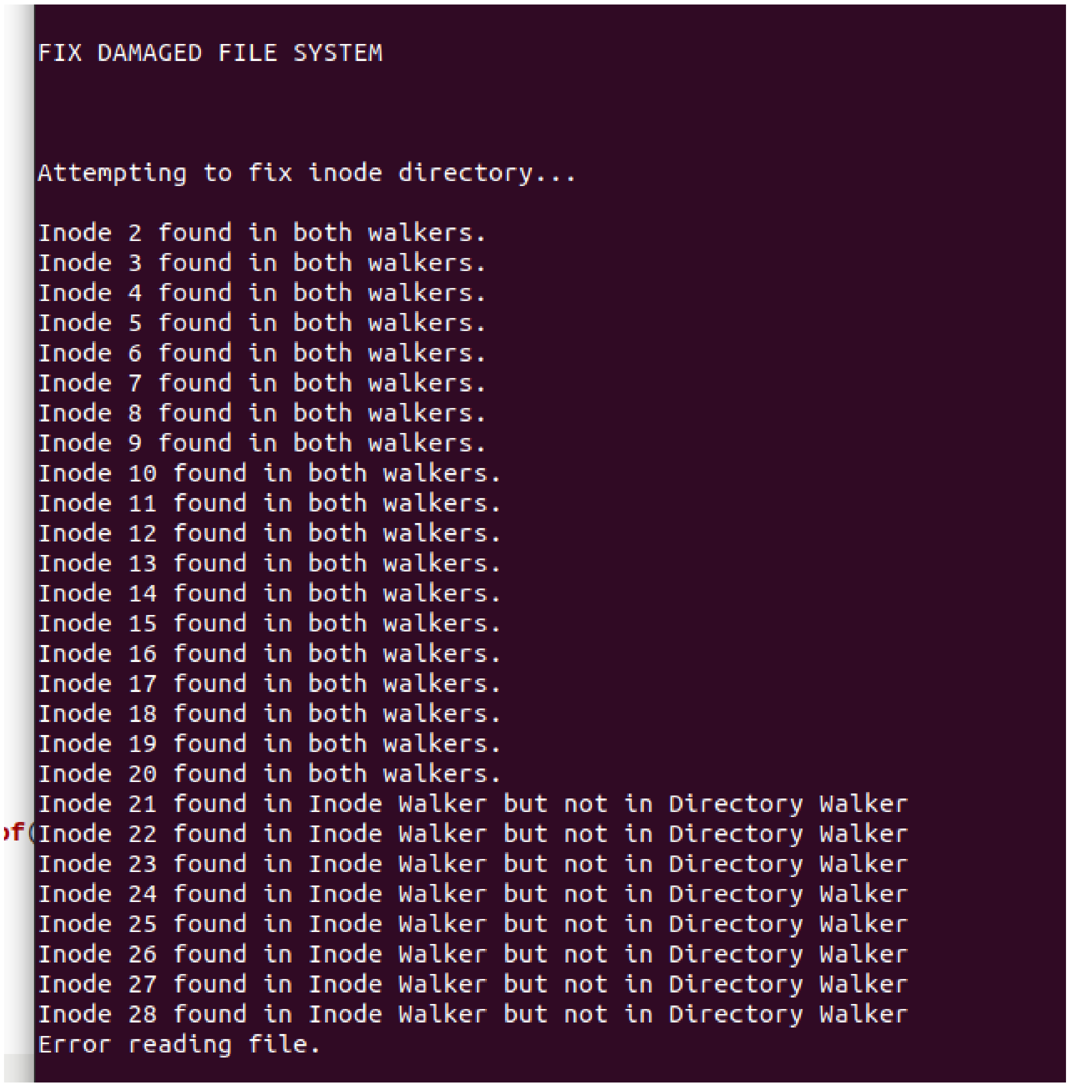


```
WALK DIRECTORY

--------------------/..
.               1
..              1
README          2
cat             3
echo            4
forktest        5
grep            6
init            7
kill            8
ln              9
ls              10
mkdir           11
rm              12
sh              13
stressfs        14
usertests       15
wc              16
zombie          17
test1           18
console         19
testfolder      20
--------------------/testfolder
.               20
..              1
test0.txt       21
test1.txt       22
test2.txt       23
test3.txt       24
test4.txt       25
test5.txt       26
test6.txt       27
test7.txt       28
--------------------
```



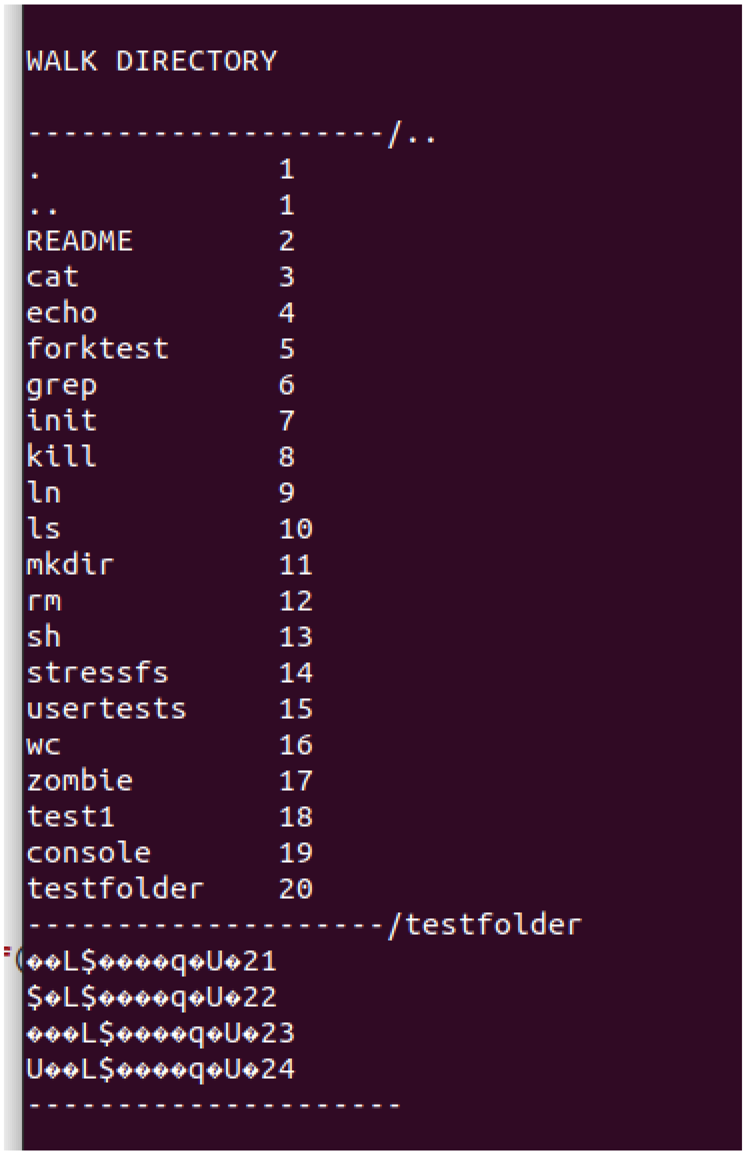
```
WALK INODE TABLE

INODE 2  ->  type: 2 size: 2287
INODE 3  ->  type: 2 size: 16420
INODE 4  ->  type: 2 size: 15272
INODE 5  ->  type: 2 size: 9584
INODE 6  ->  type: 2 size: 18640
INODE 7  ->  type: 2 size: 15860
INODE 8  ->  type: 2 size: 15300
INODE 9  ->  type: 2 size: 15156
INODE 10  ->  type: 2 size: 17784
INODE 11  ->  type: 2 size: 15400
INODE 12  ->  type: 2 size: 15376
INODE 13  ->  type: 2 size: 28012
INODE 14  ->  type: 2 size: 16292
INODE 15  ->  type: 2 size: 67396
INODE 16  ->  type: 2 size: 17152
INODE 17  ->  type: 2 size: 14968
INODE 18  ->  type: 2 size: 16320
INODE 19  ->  type: 3 size: 0
INODE 20  ->  type: 1 size: 160
INODE 21  ->  type: 2 size: 4
INODE 22  ->  type: 2 size: 4
INODE 23  ->  type: 2 size: 4
INODE 24  ->  type: 2 size: 4
INODE 25  ->  type: 2 size: 4
INODE 26  ->  type: 2 size: 4
INODE 27  ->  type: 2 size: 4
INODE 28  ->  type: 2 size: 4
```

Initial walks



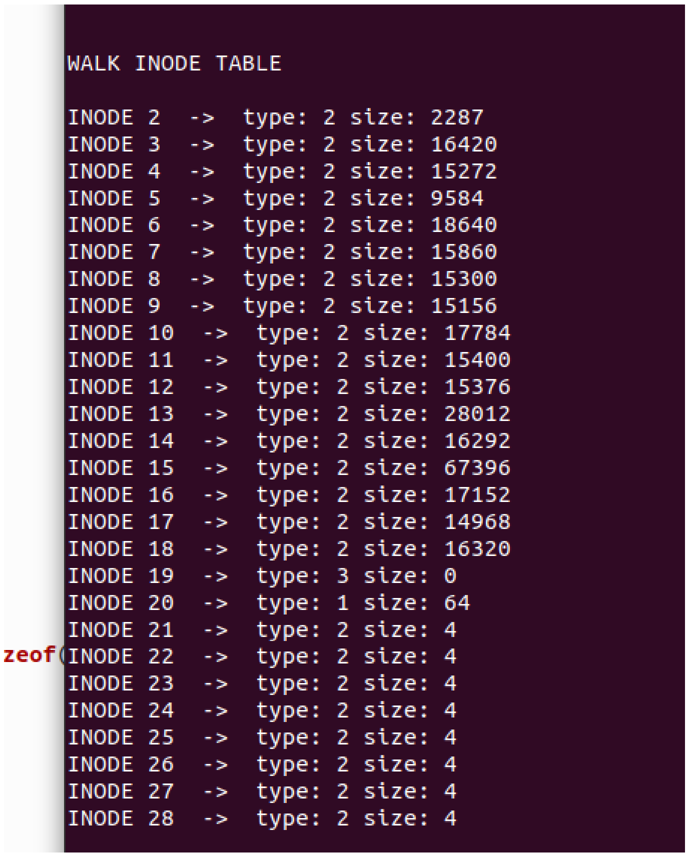
```
FIX DAMAGED FILE SYSTEM



Attempting to fix inode directory...

Inode 2 found in both walkers.
Inode 3 found in both walkers.
Inode 4 found in both walkers.
Inode 5 found in both walkers.
Inode 6 found in both walkers.
Inode 7 found in both walkers.
Inode 8 found in both walkers.
Inode 9 found in both walkers.
Inode 10 found in both walkers.
Inode 11 found in both walkers.
Inode 12 found in both walkers.
Inode 13 found in both walkers.
Inode 14 found in both walkers.
Inode 15 found in both walkers.
Inode 16 found in both walkers.
Inode 17 found in both walkers.
Inode 18 found in both walkers.
Inode 19 found in both walkers.
Inode 20 found in both walkers.
Inode 21 found in Inode Walker but not in Directory Walker
Inode 22 found in Inode Walker but not in Directory Walker
Inode 23 found in Inode Walker but not in Directory Walker
Inode 24 found in Inode Walker but not in Directory Walker
Inode 25 found in Inode Walker but not in Directory Walker
Inode 26 found in Inode Walker but not in Directory Walker
Inode 27 found in Inode Walker but not in Directory Walker
Inode 28 found in Inode Walker but not in Directory Walker
Error reading file.
```

After damage has been done to directory

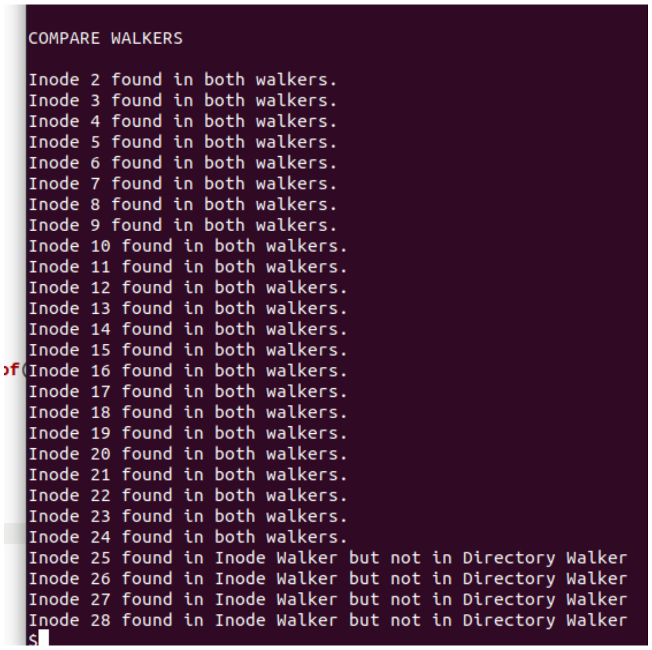


```
WALK DIRECTORY

--------------------/..
.             1
..            1
README        2
cat           3
echo          4
forktest      5
grep          6
init          7
kill          8
ln            9
ls            10
mkdir         11
rm            12
sh            13
stressfs      14
usertests     15
wc            16
zombie        17
test1         18
console       19
testfolder    20
--------------------/testfolder
��L$����q�U�21
$�L$����q�U�22
���L$����q�U�23
U��L$����q�U�24
---------------------
```

After files have been restored to the folder



```
WALK INODE TABLE

INODE 2  ->  type: 2 size: 2287
INODE 3  ->  type: 2 size: 16420
INODE 4  ->  type: 2 size: 15272
INODE 5  ->  type: 2 size: 9584
INODE 6  ->  type: 2 size: 18640
INODE 7  ->  type: 2 size: 15860
INODE 8  ->  type: 2 size: 15300
INODE 9  ->  type: 2 size: 15156
INODE 10  ->  type: 2 size: 17784
INODE 11  ->  type: 2 size: 15400
INODE 12  ->  type: 2 size: 15376
INODE 13  ->  type: 2 size: 28012
INODE 14  ->  type: 2 size: 16292
INODE 15  ->  type: 2 size: 67396
INODE 16  ->  type: 2 size: 17152
INODE 17  ->  type: 2 size: 14968
INODE 18  ->  type: 2 size: 16320
INODE 19  ->  type: 3 size: 0
INODE 20  ->  type: 1 size: 64
INODE 21  ->  type: 2 size: 4
INODE 22  ->  type: 2 size: 4
INODE 23  ->  type: 2 size: 4
INODE 24  ->  type: 2 size: 4
INODE 25  ->  type: 2 size: 4
INODE 26  ->  type: 2 size: 4
INODE 27  ->  type: 2 size: 4
INODE 28  ->  type: 2 size: 4
```



```
COMPARE WALKERS

Inode 2 found in both walkers.
Inode 3 found in both walkers.
Inode 4 found in both walkers.
Inode 5 found in both walkers.
Inode 6 found in both walkers.
Inode 7 found in both walkers.
Inode 8 found in both walkers.
Inode 9 found in both walkers.
Inode 10 found in both walkers.
Inode 11 found in both walkers.
Inode 12 found in both walkers.
Inode 13 found in both walkers.
Inode 14 found in both walkers.
Inode 15 found in both walkers.
Inode 16 found in both walkers.
Inode 17 found in both walkers.
Inode 18 found in both walkers.
Inode 19 found in both walkers.
Inode 20 found in both walkers.
Inode 21 found in both walkers.
Inode 22 found in both walkers.
Inode 23 found in both walkers.
Inode 24 found in both walkers.
Inode 25 found in Inode Walker but not in Directory Walker
Inode 26 found in Inode Walker but not in Directory Walker
Inode 27 found in Inode Walker but not in Directory Walker
Inode 28 found in Inode Walker but not in Directory Walker
$
```

Final walkthroughs