Design Document
Project3 : File System Tools
CS551, Spring 2016

Conghao Liu
Guangxin Guo
Kenji Kuramochi
04/22/2016(Fri)

First of all, the followings are the files we modified and added. Since there is a middle layer called “VFS” between a user and MFS, one can’t call the functions in MFS directly. In order to call some function in MFS, one needs to add a system call to MFS and then call it from VFS.

- /usr/include/repair.h
- /usr/include/dirwalker.h
- /usr/src/include/minix/vfsif.h

vfsif.h

```
#define REQ_WALKER        (VFS_BASE + 33)
#define REQ_BITMAPDAMAGER (VFS_BASE + 34)
#define REQ_BITMAPFIXER   (VFS_BASE + 35)

#define NREQS                      37
```

- /usr/src/servers/vfs/repair.c
- /usr/src/servers/vfs/damage.c
- /usr/src/servers/vfs/proto.h
- /usr/src/servers/vfs/table.c

proto.h

```
/* repair.c */
int do_inodemapwalker();
int do_zonemapwalker();
int do_directorywalker();
int do_bitmapdamager();
int do_bitmapfixer();
```

table.c

```
do_bitmapfixer,      /* 103 = inodebitmapfixer */
do_bitmapdamager,    /* 105 = bitmapdamager    */
do_directorywalker,  /* 106 = directorywalker  */
do_inodemapwalker,   /* 108 = inodewalker      */
do_zonemapwalker,    /* 109 = zonemapwalker    */
```

- /usr/src/servers/mfs/misc.c
- /usr/src/servers/mfs/proto.h
- /usr/src/servers/mfs/table.c

misc.c

```
fs_walker()
fs_bitmapdamager()
fs_bitmapfixer()
```

proto.h

```
/* repair.c */
int fs_walker();
int fs_bitmapdamager();
int fs_bitmapfixer();
```

table.c

```
fs_walker,          /* 33  */
fs_bitmapdamager,   /* 34  */
fs_bitmapfixer,     /* 35  */
```

1. *A manual page for each tool*

NAME

fs_walker()

DESCRIPTION

This tool acts as three utilities, “inodemapwalker“, “zonemapwalker“, and “directorywalker“. We can choose which utility can be used when you run the demo file.

name

inodemapwalker

description

The “inodemapwalker” reads the inode bit map and returns all the inodes that have been allocated. If this utility could read the inode bit map correctly, it returns ‘0’.

name

zonemapwalker

description

The “zonemapwalker“ reads the zone bit map and returns all the zones that have been allocated. If this utility could read the zone bit map correctly, it returns ‘0’.

name

directorywalker

description

We use ftw() function to traverse the given directory.Each time we receive an inode number,we pass it to directorywalker in mfs,the directory walker will print out all the zones it has.

NAME

fs_bitmapdamager()

DESCRIPTION

This tool will allow us to damage the bit map.

NAME

fs_bitmapfixer()

DESCRIPTION

This tool acts as two utilities, “fix inode bitmapblock“, and “fix zone bitmapblock“. We can choose which utility to be used when you run the demo file.

name

fix inode bitmapblock

description

The “fix inode bitmapblock“ fixes when the inode bitmap block was corrupted. If this utility could fix the bitmap correctly, it returns ‘0’.

name

fix zone bitmapblock

description

The “fix zone bitmapblock“ fixes when the zone bitmap block was corrupted. If this utility could fix the bitmap correctly, it returns ‘0’.

2. ***Design of Tools***

- inodemapwalker <fs_walker()>
    1. Load bitmap blocks one by one from disk
    2. Use bitwise operations to check each bit’s value
    3. Print out all 1’s number

- zonemapwalker <fs_walker>
    1. Load bitmap blocks one by one from disk
    2. Use bitwise operations to check each bit’s value
    3. Print out all 1’s number

- directorywalker <fs_walker>
    1. User must specify which inode he wants to check
    2. Load inode from disk
    3. Print out all the zones the inode has

- fs_bitmapdamager()

1. Check which bit is to be set, and which bitmap is to be manipulated
2. Load corresponding bitmap block to memory
3. Set the corresponding bit to the specified value

- fix inode bitmapblock <fs_bitmapfixer()>
    1. Read each inode from disk
    2. Check i_nlinks
    3. If i_nlinks > 0 and the corresponding bit is 0,set it to 1
    4. If i_nlinks = 0 and the corresponding bit is 1,set it to 0

- fix zone bitmapblock <fs_bitmapfixer()>
    1. Erase the entire zone bit map
    2. Check each inode's zones
    3. Set the corresponding bit in zone bitmap to 1

3. ***Discuss how to recover from the following situations***

- Part of the superblock is damaged

  Since some data on the superblock can be both on disk and in memory, we may be able to recover this situation by using the data on disk. Some data could be the following. -- Number of i-nodes, Number of i-node bitmap blocks, Number of zone bitmap blocks, First data zone, Padding, Maximum file size, Number of zones, Magic number, padding, Block size(bytes), and FS sub-version.

- The inode of a file is damaged

  For recovering for a inode file damage situation,we can use fsck file to check it.For example ,we can check the files in the directory of /dev/myfilelv. With the root authority, we can use the command:

  fsck /dev/myfilelv,

  to check all the file consistency.Fsck file will check the count of used inode of files and free inode files .If the files consistency is not correct,the command above will notice you that the we need to fix them.Fsck command will make them to get fixed. If it's not worked,we have to recover from backup.

  mkfs /dev/myfilelv

  mount /dev/myfilelv /myfilesys

  cd /myfilesys

  restore -r

- The inode of a directory file is damaged

  The name and inode number pairing in a Minix directory is the only connection between a name and the thing it names on disk. The name is kept separate from the data belonging to the thing it names (the actual inode on disk). If a disk error damages a directory inode, file data is not usually lost; since, the actual data for the things named in the directory are stored in inodes separate from the directory itself. If a directory is damaged, only the names of the things are lost and the inodes become “orphan” inodes without names. The storage used for the things themselves is elsewhere on disk and may be undamaged. We can run a file system recovery program such as fsck to recover the data (but not the names).

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

  We would be able to recover it if and only if we have a backup of that block