Project Design Document

Team Number: *#5*

Group members: *Xiaowei An, Sachin Ambalkar, Ying Ma*

Project Name: *File System Tools*

Date : *04/25/2015*

1. Introduction:

1.1 Purpose:

Comprehend and grasp the conception about File System in Minix3 by implementing File System Tools.

1.2 Requirements (by coding):

This File System Tools are able to fix the below problems of damage in File System:

A. The directory file is damaged

B. The inode of directory is damaged

C. The inode and the directory file are both damaged

1.3 Requirements (by answering questions)

The questions that could we recover the data in below different cases is discussed:

A. Part of the Super Block is damaged

B. The inode map is damaged

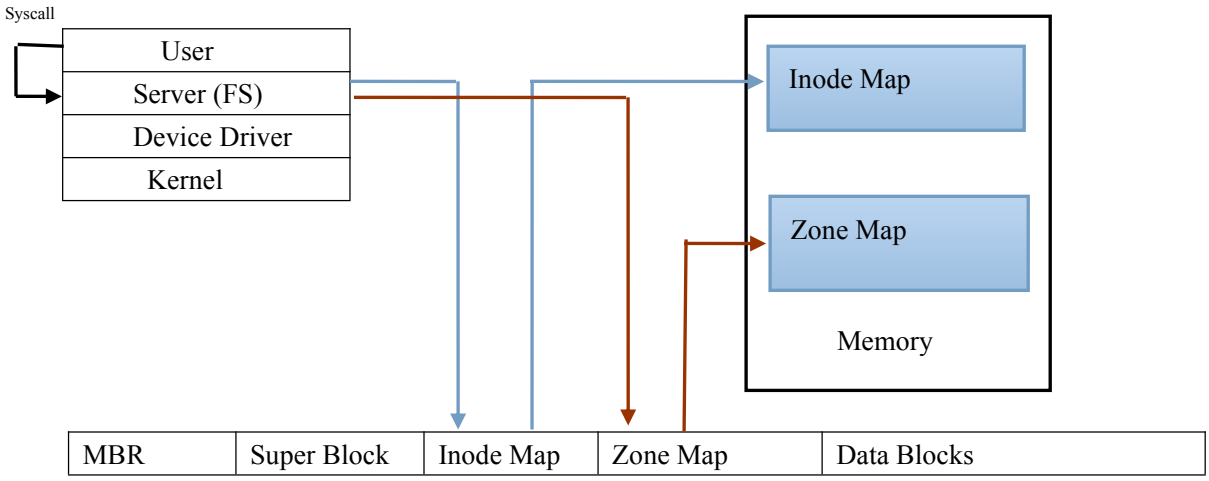
C. The zone map is damaged

D. The inode of a file is damaged

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

2. System Architecture:

We access the inode table and zone table by the system call:

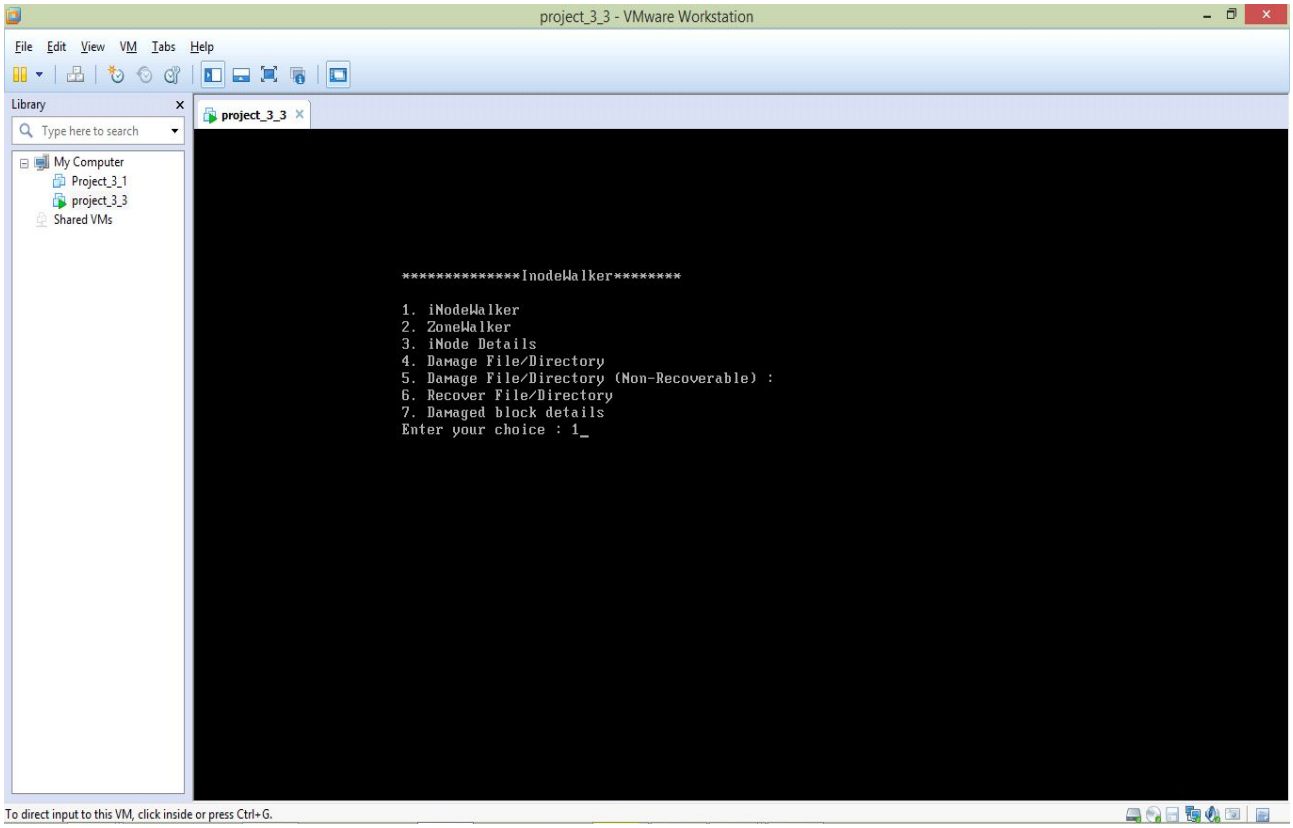


3. Algorithm:

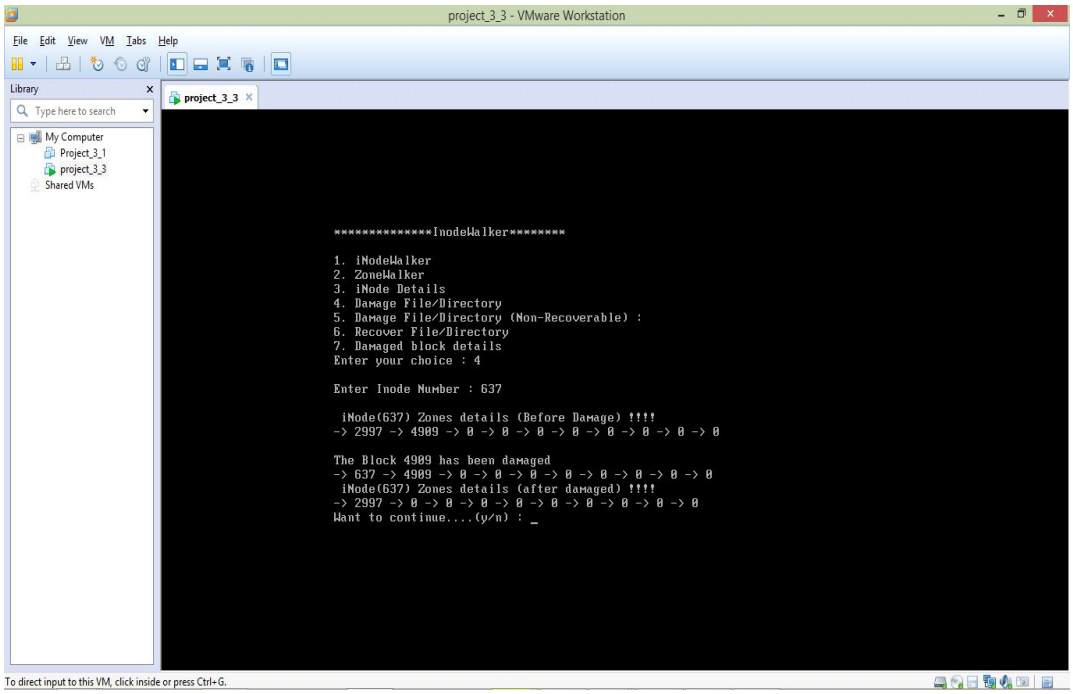
Firstly we clear the info of zone in one inode.
After we close the file, this inode will be flushed in disk because it is dirty.
Then we cannot access the directory becuase of the corrupted zone data.
However, in the zone map, all the bits that belong to this inode are keep still.
So we just compare the two maps.
According the Inode map, we use the get_inode in inode.c to scan every inode. For each inode, we can get the zone info, so that we can compare with the zone map one by one. At last, we can find some bits in zone map are 1, but no inode contains them, which means these zones belong to the inode.
Finally, we just re-fill the zone info of the inode.

4. Result by screen shot:

Firstly we enter the user menu interface:

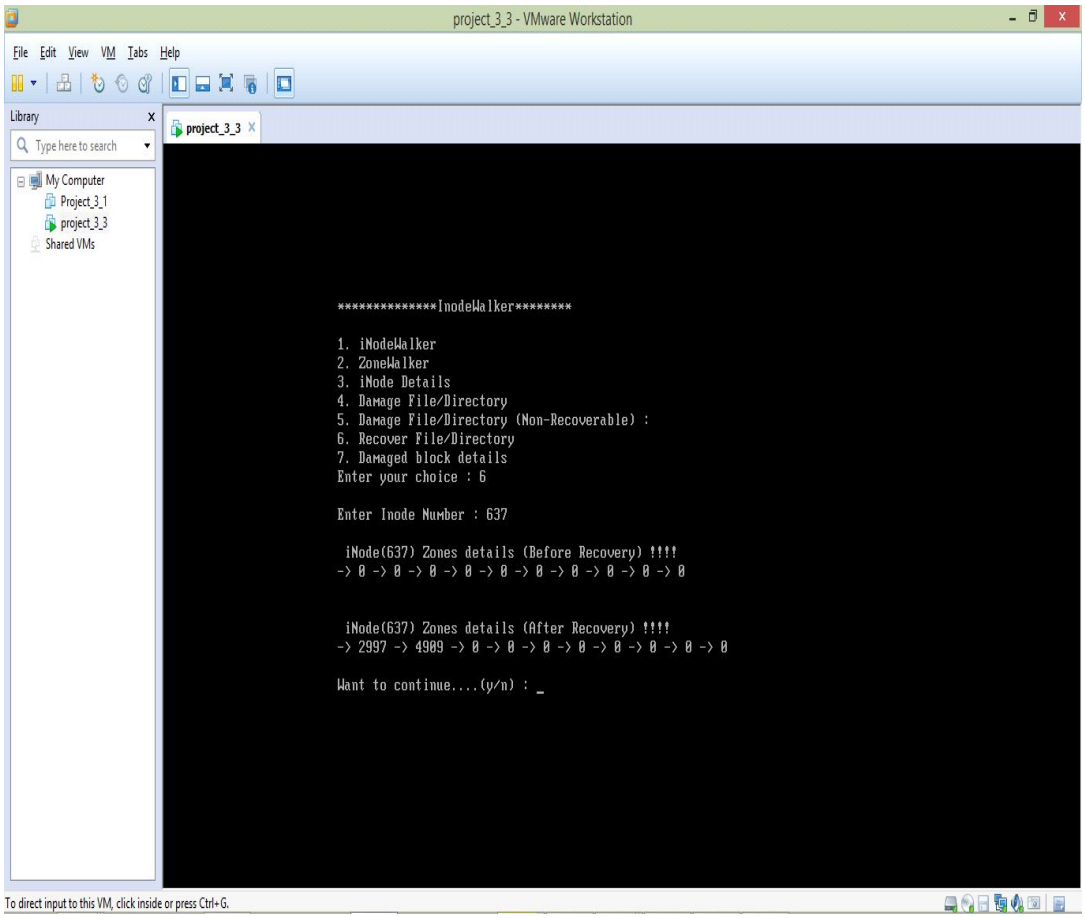


Secondly we choose a directory and clear its zone



At this time, we cannot access the directory.

Choose the option 6, which is used for recovery.



Finally we can access the directory.

5. Question Discussion:

The questions that could we recover the data in below different cases is discussed:

5.1. Part of the Super Block is damaged

In this case, we have to analyze it from 2 aspects.

Firstly, for prevent the corruption of super block, we can use backup.

E.g:

Original Format:

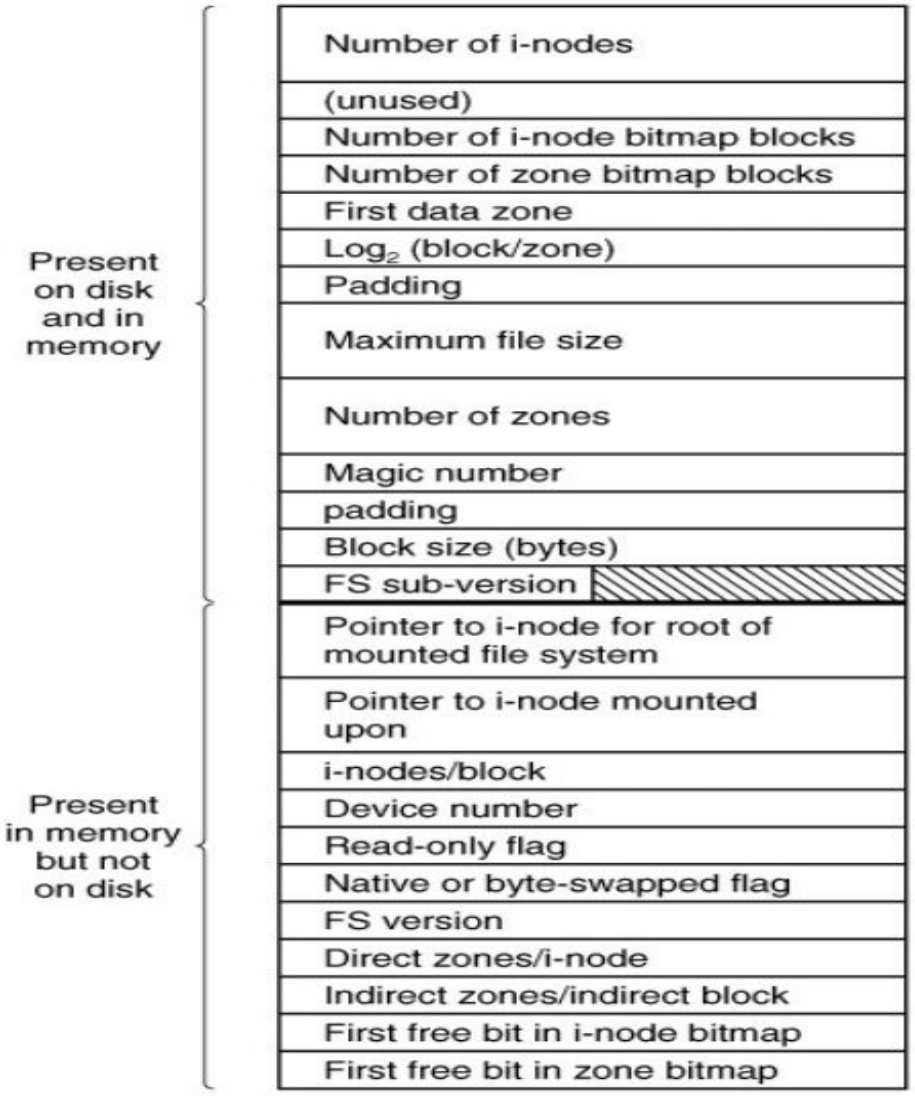
MBR	Super Block	Inode Map	Zone Map	Data Blocks

Adding a backup for Super Block:

MBR	Super Block	SB backup	Inode Map	Zone map	Data Blocks

So at once the code in MBR find the first super block is damaged, it will read the backup.

The second method, we consider to re-build the Super Block by calculation.



In fact, not all the items in this structure are necessary. We can re-build some of them by other items. We will discuss all the situations one by one (present on disk

A. Number of Inode

It is crucial value. It cannot be damaged.

B. Number of Inode Bitmap Blocks

We can recover it by number of Inode.

C. Number of Zone Bitmap Blocks

We can recover it by number of zones

D. First Data Zone

We can recover it by calculation how many blocks located before the data blocks

E. Log2

At minix3, currently, we know that this value is 1. Of course in the different OS, we need to discuss it.

F. Maximum file size

It is necessary.

G. Number of zones

We know the first data zone, and the zone size, we can recover it.

H. Magic number

It is necessary.

I: Block Size

We can calculate it by the value of log2

J: FS sub-version

It is not necessary, we can check the OS version and FS version in internet then re-fill it,

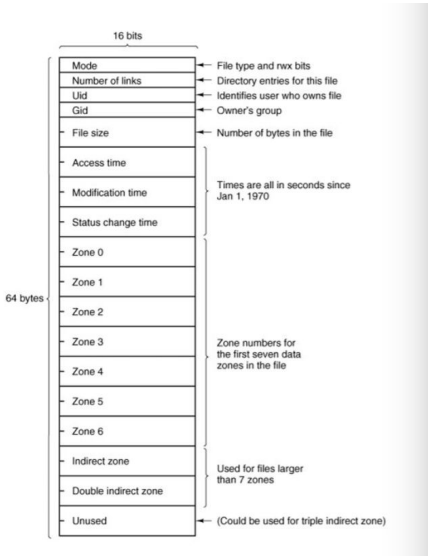
5.2 The inode map is damaged

We have to consider the situation of inode blocks. We know the size of inode, and the number of inodes, so we can read it one by one. So the point is that if we can distinguish certain inode is used or free? However, once file system free a inode, the free_inode function will reset the bitmap immediately. So this inode in the inode block maybe contain dirty data, that means we don't know it is used or free.

5.3 The zone map is damaged

We can re-build it by scan all the inodes

5.4 The inode of a file is damaged



A. Model
   It can be rebuilt. Because it is not necessary
B. Number of links
   It is necessary
C. Uid & Gid
   In fact, if we don't care the security, it is not important, which means we can reset it.
D. File Size
   It is necessary
E. Access time & Modification time & Status change time
   Id depends. If our goal is opening it, these three value are not important.
F. Zone
   Of course, we can know this inode contains which zones. However, we don't know the order. So we cannot re-build it.

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

We cannot fix them normally. However, we can use some ECC to cover some bits shift.