Ext2 File System Analyzer

Bin Chen
School of Information Science and Technology
Shanghaitech University
Student ID: 25114447
Email: chenbin@shanghaitech.edu.cn

Yimo Chen
School of Information Science and Technology
Shanghaitech University
Student ID: 78645321
Email: chenym@shanghaitech.edu.cn

1. Introduction

As a course project of *Operating System II*, we design a Ext2 File System Analyzer. As we've known, FAT&NTFS have already had developed analyzer, like xuetr. However, no one has developed an anti-rootkit analyzer for Linux Ext File System. So, we are going to design an Ext File System Analyzer starting up from Ext2. As a result, we can now use our analyzer to have access to an Ext2 File System directly without system call and we can try to recover deleted files.

2. Ext2 File System

2.1. Brief Introduction

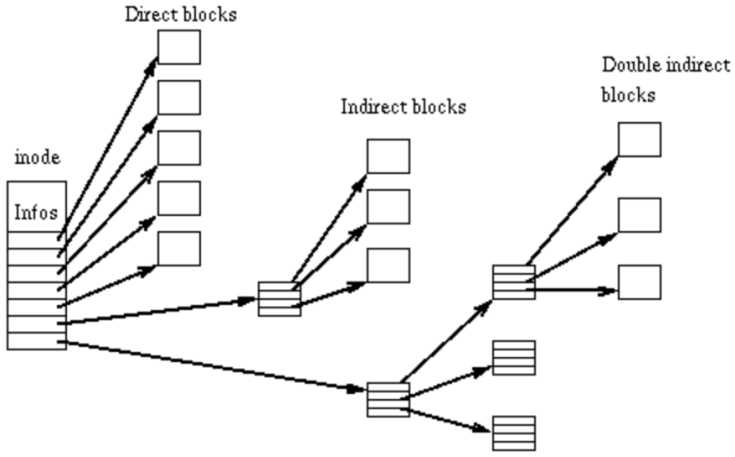
The second extended filesystem is a Linux kernel file system which has been designed according to the same principles as BSD.

2.2. Data Structures

In ext2 filesystem, space is split up into blocks and grouped into block groups. Data for any given file is typically contained within a single block group where possible. Each block group contains a copy of the superblock and block group descriptor table, and all block groups contain a block bitmap, an inode bitmap, an inode table, and finally the actual data blocks.

2.2.1. Inodes.

Every file or directory in ext2 filesystem is represented by an inode(index node). Data of the file or directory is included by inode, such as size, permission, ownership and location on disk.



2.2.2. Directories.

In ext2 filesystem, each directory is a list of directory entries which represented by one file name and inode number and consists of the inode number, the length of the directory entry length, the length of file name, the file type, and the actual text of the file name.

To find a file, ext2 filesystem just searched front-to-back for associated filename in directory list, each search skip the length of current directory entry length to find the next active directory entry(some directory entry still alive but inactive because they are deleted).

The directory entry for subdirectories are implemented same as file, so find a subdirectories in the totally same as find a file. The principle of hard links is just one inode with different filename, so you may access same file by same inode but different filename.

The root directory is always stored in inode number two, so ext2 filesystem can find root directory by directly access that one associate inode two. There are two special directories store in each directory in ext2 filesystem. The current directory and the parent directory are implemented by storing the names "." and ".." in the directory, and the inode number of the current directory and the parent directory are in the inode field. These two special directories

are automatically created when any new directory is made, and can't be deleted during current directory active. [1]

3. Implementation

3.1. Build-up

We create a new ext2 file system with blocksize==1 with the following command:

$ dd if=/dev/sda4 of=bean3 bs=1k count=64000

Then, we mount the file system bean3 on loop0 to get a graphic access to it.

$ losetup /dev/loop0 bean3
$ mount -t ext2 /dev/loop0 /mnt/extfs

3.2. Logic

Given the fact that the inode for the root of the file system is *2*, we can use inode as a key to have access to every file in the file system.

As a result, if we can get all the inode in the file system, we can use inode as a pointer to find out what is stored in that inode. Normally, we use rec_len to find the next inode position. However, given the name_len, we can find the next inode manually. For instance, if name_len is smaller than 4, the length of the name will occupy 4 bytes and if name_len is 5 to 8, the length of the name will occupy 8 bytes, as shown in figure 1. So, given a inode, we can find out all the existing and deleted inode that is in the given inode. Using this method, we can try to recovery the deleted file if we can still find the inode.

3.3. Function

We now implement the ls, cat, backup, del, exit function for the analyzer.

ls —- the key function to list files and directories. we can use *$ ls all inode_number* to check if there is some hidden files or deleted files.

cat —- to browse the file by using inode.

backup —- to backup a file using inode. If you find out a inode of a deleted file, you can simply using *$ backup inode_number* to recovery the deleted file.

del —- to delete a file using inode.

4. Future Work

We may keep on developing this analyzer with some more functions.
Some anti-rootkit functions like list out those hidden files that is placed by cutting the inode table.

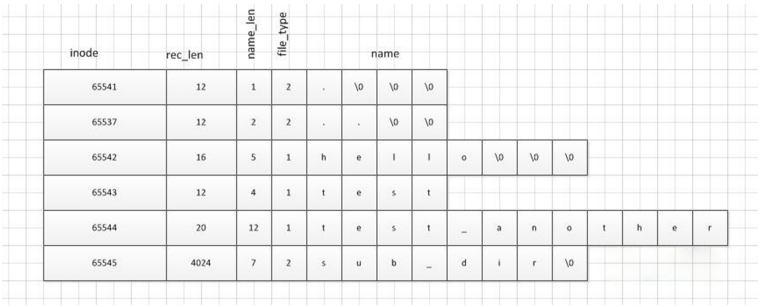


Figure 1. Implementation Example

5. Conclusion

During this project, we learned the detail of ext2 file system and recognize its strengths and weaknesses. Through the principle of ext2 file system, we designed a way to recover the deleted data on it. As a result, we implement the way which discussed in previous part in Rust language. The source code of our program can find in github [2].

References

[1] Wikipedia, "ext2," https://en.wikipedia.org/wiki/Ext2.

[2] C. Bin and C. Yimo, "Ext2-fs-analyzer," https://github.com/TCMOOO/Ext2-fs-Analyzer.