Middle East Technical University Department of Computer Engineering

CENG 334

Introduction to Operating Systems

Spring 2017-2018

HW3

Due date:29th May 2018, 23:55

1 Objectives

This assignment aims to get you familiar with file system structures by an implementation of a simple file recovery tool. Your application will list deleted regular files and recover if possible. Recovered files should be moved to lost+found directory. You are not expected to find the names of the deleted files, you will assign a generic name (file01, file02, ...) to each recovered file and you can assume the number of files will be smaller than 100.

Your program should find the deleted regular files from the inode table. Find the blocks used by each file. Determine whether a block is reused by another inode. If a deleted file's blocks are still available inode should be modified to a usable state and file should be recorded under lost+found directory.

If a block is marked as occupied then it is in use by an active inode. If there is a block that is owned by more than one file and both of them are deleted than newer file should be recovered.

2 EXT2

Following diagram shows the structure of a ext2 system on disk. The first 1024 bytes of the disk is always reserved as a boot block. After the boot block, a number of block groups exist. Each block group starts with its super block, then a number of group descriptors are placed. A bitmap of existing blocks and a bitmap of existing inodes occupies one block of disk each. The size of the inode table that comes after the inode bitmap depends on the number of inodes created while formatting. The rest of the block group consists of data blocks.

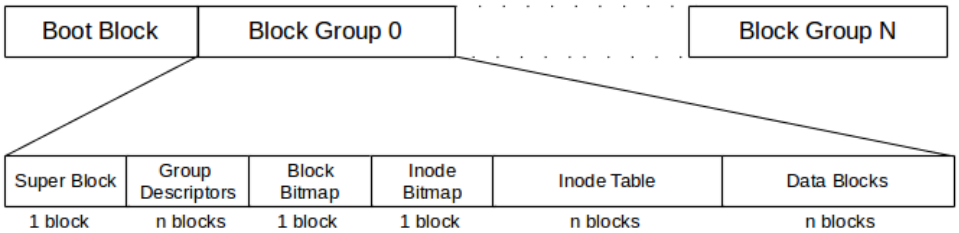

Figure 1: EXT2 structure

For more information on EXT2 file system, you should look at OSDev wiki page on EXT2. Also this page by Dave Poirer contains more details.

But important details are:

- Inode and block numbering starts at 1.
- Block numbering starts at the beginning of the disk. Super block of the first group resides in block 1.
- The root inode always resides in inode number 2.
- The first 11 inodes are reserved.
- There is always a lost+find directory in root and the 11^{th} inode is the lost+found directory.
- Disk sectors are 512 bytes.

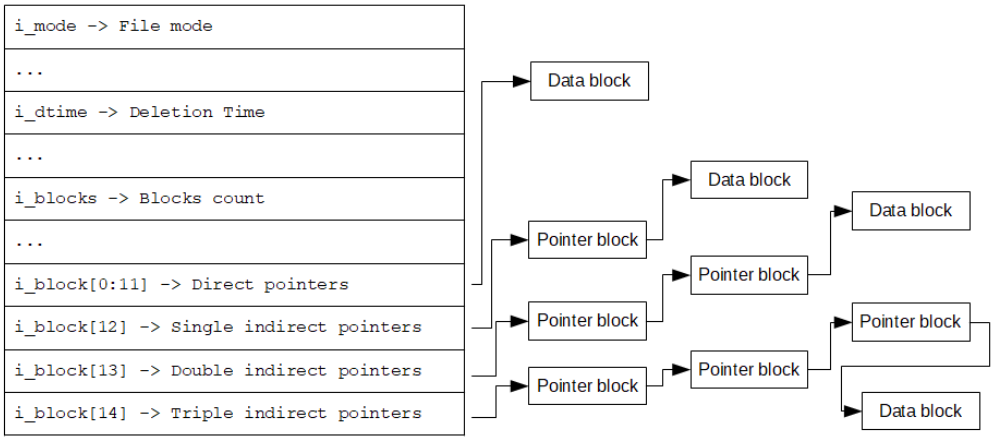

Figure 2: inode structure

2.1 Details

You can create a disk image with 128 blocks of size 1024 with the following command:

```
$ dd if=/dev/zero of=image.img bs=1024 count=128
```

The created disk image can be formatted with `mke2fs`. Following commands formats the disk and forces the creation of 32 inodes.

```
$ mke2fs -N 32 image.img
```

You **have to** use FUSE based `fuseext2` to mount your image to a folder you own.

```
$ mkdir mnt
$ fuseext2 -o rw+ image.img mnt
```

and unmount with:

```
$ fusermount -u mnt
```

If you use standard `mount` command, the erase behaviour removes the block numbers from inodes, so you can not recover the files. Also do not mount and modify the image you have created with the GUI (file managers, Nautilus in Ubuntu). Only use `fusermount` and modify the image file from the console.

In this homework, you are **not** expected to handle more then one block group. But different block sizes and different number of inodes and different number of blocks must be handled.

You can assume that there will not be any files in the lost+found directory.

You can inspect differences between two image files using a `xxd` hex dump and `diff`. Following command is an example for this purpose.

```
$ diff <(xxd image1.img) <(xxd image2.img) > images.diff
```

2.2 Implementation, Compilation & Execution Details

You should look at `hw3.c` file for some ideas for your implementation. Also given header file `ext2.h` has helpful comments. In your recovery tool, only files that are not overwritten expected to be recoverable. You should scan all inodes, find the ones listed as deleted, determine the blocks used by each deleted file and recover those which are not overwritten.

You should modify the inode of the recovered files to marked as a regular file and has at least user read privileges (`inode.i_mode = EXT2_S_IFREG | EXT2_S_IRUSR`).

Any standard library can be used in your code. You **have to** provide a makefile with your implementation which creates an executable named `recover`.

Your code will be executed with `./recover imagefile`. You can assume a valid, uncorrupted ext2 image file. Your program have to give the following output on standard output during recovery.

1. A list of files that marked as deleted that is generated from inode structures.
2. A list of files that recovered and moved to lost+found.

An example output:

```
$ ./recover imagefile
file01 0000001 1
file02 0000002 1
file03 0000003 1
###
file01
file03
```

Example shows three files where two of which is recovered. First part of the output lists the files with its deletion time and numuber of blocks. Second part lists the files that are recovered. The names will be assigned by your program, you are not expected to recover the names of the files, only the contents are expected. In this example, blocks of `file02` are overwritten by either current files or one of the deleted files that is recovered.

3 Regulations

1. **No Late Submission**
2. Your code have to be in C/C++.

3. Submission will be done via COW. Create a tar.gz file named *hw3.tar.gz* that contains all your source code files and a makefile. The executable should be named *recover*.

4. Following sequence of commands should compile and run you code, otherwise you lose 10 points.

```
$ tar -xf hw3.tar.gz
$ make all
$ ./recover
```

5. Your codes will be evaluated with a black-box approach and have to compile and run on lab machines.

6. Please ask your questions related to the homework on piazza instead of email. Your friends, who may face with same problem, can see your questions and answers.

7. Do not cheat.